Performance Bounds on Sensor Placement Algorithms and Online Outlier Detection in the Power Grid

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRICAL ENGINEERING

December 2016

By

Muhammad Sharif Uddin

Dissertation Committee :

Anthony Kuh, Chairperson

Aleksandar Kavcic

Anders Høst-Madsen

Matthias Fripp

Kyungim Baek

To my parents

Acknowledgements

I am deeply indebted to many extraordinary individuals for making my graduate studies one of the most rewarding periods of my life. First of all, I would like to express my deepest gratitude to my advisor Prof. Anthony Kuh for his guidance and support throughout the course of my doctoral studies. I have greatly benefited from many insightful and inspiring discussions with him on many topics in machine learning and signal processing. I am sincerely grateful for the freedom he accorded me to pursue new research directions that motivated me to widen my research perspective.

I would also like to extend special thanks to Prof. Aleksandar Kavcic for his thoughtful guidance during the early periods of my doctoral studies and later serving on my dissertation committee. My many discussions with him have had great influence on my research work. I would like to thank Prof. Anders Høst-Madsen, Prof. Mathias Fripp, and Prof. Kyungim Baek for taking the time out of their busy schedules to serve on my dissertation committee and providing insightful comments on my work. I would further like to express my gratitude to Prof. Reza Ghorbani for his support during my studies.

I would like to express gratitude to Prof. Toshihisa Tanaka of Tokyo University of Agriculture and Technology, Japan for his valuable suggestions during his visit to UHM and during my visit to Japan. I would also like to thank Prof. Kazushi Ikeda of Nara Institute of Science and Technology, Japan for offering me the opportunity to work with his research group. Furthermore, I would like to thank Prof. Vijay Gupta of the University of Notre Dame, Prof. Marija Ilić of Carnegie Mellon University and her then PhD student Yang Weng (currently a Postdoctoral scholar at Stanford University) for their time and useful discussions during my visits to the respective institutions. They have greatly influenced me in my research work.

Special thanks to my fellow students from the department, especially to the lab mates in Holmes 490 : Navid, Jeremy, Seyyed, Ciril, Minqi, and Trevor, for their encouragement and

support during my years of study. I would also like to take this opportunity to thank the faculty members and EE office staff, current and former, for their help and support. I am grateful for the opportunity I was given at the department to learn and grow.

Finally, I would like to thank my mother, late father and my siblings for their immense support and encouragement throughout my life. I am also deeply grateful to my wife Hannah for her constant support and encouragement during my graduate studies.

Abstract

Accurate real-time estimation of the states is of critical importance in the operation of the power grid. The quality of the state estimates is essentially dependent on ensuring the collection of measurement data that provide maximal information about the states and ensuring the integrity of the collected data. This thesis addresses the problem of ensuring quality of the suboptimal solutions to the optimal sensor placement problem and also presents algorithms for detecting outliers or malicious data in the power grid. In the sensor placement problem, the number of sensors that can be deployed is often limited by costs and other resource constraints. Finding the best subset of sensor locations in a large network is prohibitively complex, forcing us to look for suboptimal algorithms. In this thesis we obtain numerical bounds on the suboptimal algorithms to assess the performance of these algorithms in the absence of the optimal solution. Given noisy measurements and knowledge of the state correlation matrix, we use the linear minimum mean squared error estimator as the state estimator to formulate the sensor placement problem as an integer programming problem. We develop a set of approximate algorithms and derive a set of analytical nested performance upper bounds to the optimal solution based on the structure of the data correlation matrix.

The second part of the thesis investigates algorithms to ensure data integrity by detecting outliers in the sensor data. We study the detection of gross measurement errors and hidden data attacks in the power system as an online outlier detection problem. An online probability density based technique is presented to identify bad measurements within a sensor data stream in a decentralized manner using only the data from the neighboring buses and a one-hop communication system. Analyzing the spatial and temporal dependency between the measurements, the proposed algorithm identifies the bad data. To develop an online outlier detection algorithm with lower complexity, a sparse online least-squares one-class support

vector machine classification algorithm is developed to provide real-time quality information, before the data is fed into the computationally expensive state estimator. An approximate linear dependence cost criteria is used to obtain a sparse solution by sequentially processing each data point only once, keeping with the requirement of data processing over data stream. The performances of the proposed algorithms are then verified through simulations on IEEE benchmark test systems.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

1.1 Overview

Data gathering, data mining, or data analytics has become increasing more important in a wide range of applications from energy to health care to social networking to business to environmental research, etc. A crucial part of the data collection process is the decision as to which of the multitude of the possible measurements are to be collected. The number of sensors that can be deployed is often limited by costs and other resource constraints. However, finding the best subset of sensor location is computationally difficult, in fact NP-complete and often the sensor locations are chosen suboptimally [1].

Ensuring the integrity of the collected data is another fundamental step in data analysis. Without reliable data from the deployed sensors, the system operator's ability to respond to the current operating conditions may be severely compromised. The aim of the outlier detection algorithm is to detect data that do not conform to the patterns exhibited by the true status of the system.

The traditional power grid was designed for uni-directional power flow from the generators to the consumers. In order to take advantage of the modern technological innovations and mitigate the ever increasing energy demand, the integration of large amount of distributed energy sources (e.g., rooftop solar panels) has become very popular, resulting in bi-directional power flows in the grid. With the integration of distributed resources in the distribution grid, collecting and analyzing the data from the grid has become of critical importance for the reliable and efficient operation of the grid. Moreover, energy storage devices and plug-in electric vehicles introduce more complexity into the grid. Deployment of advanced sensor network (e.g., advanced metering infrastructure, phasor measurement units) for reliable data collection is essential to ensure stability and reliability in the operation and control of the complex grid. Consequently, the physical grid is becoming increasingly more vulnerable to malicious attacks targeting the cyber-physical infrastructure.

This thesis mainly addresses the problem of ensuring quality of the suboptimal solutions to the optimal sensor placement problem and detecting outliers or malicious data in the power grid.

1.1.1 Optimal Sensor Placement in Power Grid

State estimation (SE) is a key function in modern energy management systems, where various crucial control tasks depend on the accurate snapshots of the system state [2]. Conventional state estimators rely on the redundant measurements captured by supervisory control and data acquisition (SCADA) systems [2], which can only take non-synchronized measurements. These measurements are too infrequent to capture the dynamics of the power grid [2]. With

the advent of phasor technology, time synchronized measurements can be obtained using phasor measurement units (PMUs) [3]. These devices take advantage of the global positioning system (GPS) technology to provide time-stamped measurements of the bus voltage magnitudes and phase angles [3].

Traditional SE using SCADA measurements is nonlinear, and is solved using iterative algorithms [4]. The PMUs, on the other hand, can directly measure the states at the PMU-installed buses, and the states of all the connected buses (if enough channels are available). In fact, given the high measurement precision and reliability of the PMUs, we can consider the PMU measurements to be low-noise refinements of certain states (exactly those states that are measured by the PMUs) [3]. Since the PMUs can refine only a small subset of all state estimates, a common task is to refine the remaining state estimates (corresponding to the buses not carrying PMUs) using the sparse PMU measurements.

To measure all the state variables, the PMUs need to be installed at around one third of all the buses [5]. Since this goal is unlikely to be achieved in the near future, researchers look for the best solutions to deploy PMUs at a smaller subset of the buses, such that the state estimation error is minimized.

In this paper, we consider the optimization problem where we have n bus locations (where we can deploy PMUs) and m PMUs to place ($m << n$). We formulate the optimization problem to minimize the mean squared estimation error. Finding the optimal solution for the PMU placement problem is very difficult. In fact, it has been shown that the problem is NP-complete [5]. This means that there is no known efficient method to solve this problem with computational complexity that is polynomial in n. For this reason, heuristic approaches (e.g., greedy algorithm [6], gradient projection algorithm [7] etc.) are typically applied to search for good suboptimal solutions. But the question is, how can we guarantee a heuristic solution is close to the optimal one, when we have no computationally feasible method of computing the optimal solution? The only way to guarantee the quality of a heuristic solution is to compare it to a provable and computationally feasible performance bound.

However, for the PMU placement problem, no tight bounds are available either. Hence, in this thesis, we propose upper bounds on the optimal solution that allow us to bound the difference between optimal and suboptimal solutions.

1.1.2 Online Outlier Detection in Power Grid

An outlier is a data point which is significantly different from the remaining data. Outliers contain useful information about abnormal characteristics of the systems and entities, which impact the data generation process and recognition of such unusual characteristics provides useful application-specific insights [8]. Outlier detection is a fundamental step in data quality, management, and analysis tasks. For example, in the power grid system, a large amount of data is collected from sensors and then processed to provide a snapshot of the current system status of the grid to the operator. An outlier in the sensor data could be an indicator of faulty instruments, line faults, or false data injection attacks [4, 9, 10]. Failure to promptly detect outliers in the incoming data may compromise the operator's ability to take remedial actions.

In the power system outlier or bad data detection is mostly studied in the context of the state estimator to detect gross measurement errors. Most of these methods are based on Chi-squared test or largest normalized residual test (r^N_{max} test) [4, 11–13]. These solutions are performed offline and require multiple runs through the state estimator, thus making them unsuitable when the data arrive more frequently and real time quality information of the data are required. In order to deal with the large amounts of continuously arriving data it is necessary to be able to identify any bad data in an online fashion before the next scan of measurements arrive for processing.

In this thesis, we propose two methods for online outlier detection, namely online kernel density estimation based method and online least-squares one-class support vector machine based method, that are able to deal with large data streams and are suitable for bad data identification in the power grid.

1.2 Literature Review

1.2.1 Prior Work on Optimal Sensor Placement in Power Grid

There has been a substantial amount of previous research on the optimal placement of sensors. In Dhillon et. al. [14, 15] the optimal placement of sensors is considered where the probability of sensor detection depends upon distance with sensors placed on a two or three dimensional grid. Placement of wireless sensors was studied in [16] where the sensors are placed at nodes such that the network satisfies a predetermined lifetime and coverage requirement. In work by Krause et. al. [1] they model spatial phenomena as a Gaussian Process and consider placement of sensors again using optimal experimental design. The goal is to maximize mutual information and the problem becomes a combinatorial optimization problem that is NP-complete. This paper also discusses a greedy algorithm and shows that mutual information is submodular and is monotonically increasing for a small number of sensors. Performance bounds are obtained for the greedy algorithm. Simpler reduced computation algorithms and robust algorithms are also considered. In [17], the authors consider the sensor selection problem in a wireless sensor network for event detection, under two hypotheses - event occurring and event not occurring. They propose the maximization of the Kullback-Liebler and Chernoff distances between the probability distributions of the selected sensor measurements, under these hypotheses, as the optimization criteria. After proving that this problem is NP-hard, the authors propose a greedy algorithm as a general approach to solve this problem suboptimally. More recently, Sakiyama et. al. [18] presented an interpretation of the optimal sensor placement problem as a graph sampling problem and proposed a heuristic greedy algorithm based on the sampling theory for graph signals.

In the power systems research, optimal sensor placement is commonly studied by considering placement of Phasor Measurement Units (PMUs) [19–22]. In [19, 20] the problem is formulated as a state estimation problem with PMU placement depending on a key condition to make the system observable. In [23, 24] the optimization criterion is to maximize

the measurement redundancy while minimizing the required number of PMUs. To solve this problem, [23] considered the phasing of PMU deployment in an integer linear programming (ILP) framework, while [24] took a binary particle swarm optimization (BPSO) based approach. More recently in [6], PMU placement is considered in a different context where observability is assumed and the goal is to optimize experimental design using a different criterion. The solution involves solving an integer programming problem which is NP-complete. However, an approximate greedy solution is found that gives good results and runs in polynomial time. Then the greedy algorithm is tied to submodular and monotonic functions where bounds can be obtained to the greedy algorithm in relationship to the optimal algorithm. An estimation-theoretic approach to the PMU placement problem is proposed in [7, 25]; after posing system state estimation as a linear regression problem, a convex relaxation is developed to suboptimally solve the PMU placement problem.

More recently, researchers have also considered PMU placement with constraints considering the effects of zero-injection buses [26–28]. In [28], the authors propose a genetic algorithm (GA) while [27] proposes the ABC (Artificial Bee Colony) algorithm to solve a hybrid optimization problem. Comprehensive reviews of different criteria used in optimal PMU placement and proposed heuristic solutions are presented in [21, 22].

1.2.2 Prior Work on Outlier Detection in Power Grid

In recent years the online outlier detection problem has received significant attention in the data mining community (see [29] and the references therein). The research done in this area can be extended to power grid applications. In power systems research most outlier detection methods are based on Chi-squared test or largest normalized residual test (r^N_{max} test) that are performed as post-processing steps after the state estimation process is completed [4, 11–13]. In more recent research, the binary particle swarm optimization techniques have been proposed to identify multiple bad data [30, 31]. Asada, Garcia, and Romero formulate the bad data identification as an optimization problem in [32] and present a Tabu search meta-

heuristic solution. In [13], the authors propose a bad data detection scheme by combining two independent state estimators that process the data from PMUs and SCADA units, separately. All of these solutions are performed offline and require multiple runs through the state estimator, thus making them unsuitable when the data arrive more frequently and real time quality information of the data are required. In order to deal with the large amounts of continuously arriving data, it is necessary to be able to identify any bad data in an online fashion before the next scan of measurements arrive for processing.

A more recent concern in the power grid research community has been the data integrity attacks in the smart grid [10]. The strong coupling of the communication networks in the smart grid makes the grid particularly vulnerable to malicious data attacks. In [10], Liu et al. demonstrated that with complete knowledge of the grid topology and transmission line admittances, an attacker can design and inject malicious measurement data into the grid that will be undetectable by the r^N_{max} test in the DC state estimation model. False data attacks with a more practical assumption of limited knowledge of the grid topology was discussed in [33]. Recently, Yu and Chin showed that an attacker need not have any prior knowledge of the grid topology and, in fact, it possible to design *approximately* stealthy false measurement vectors that can bypass the r^N_{max} test using principal component analysis (PCA) approximation of the covariance matrix of the measurements [34]. However, in the AC state estimation model, the attacker needs not only access to grid topology but also estimates of the current states to design unobservable attacks [35, 36].

Several approaches have been proposed in the literature for the detection of false data injections in both DC and AC state estimation model [36–41]. The addition of secure phasor measurement units to create protected measurements to detect coordinated attack at a small number of meters has been proposed in [13, 42]. Li et al. [39] proposed a generalized likelihood ratio (GLR) based cumulative sum (CUSUM) algorithm for quickest detection of false data injection attacks in a wide area monitoring system. In [37], Esmalifalak et al. proposed a PCA based method for detection data injection attacks. They use PCA to reduce the dimension of the collected data to the principal components and fit the principal components

to a Gaussian probability model. An attack is detected when the observed statistic of the data deviates from the model fitted to the historical data. Ozay et al. [40] demonstrated the effectiveness of several machine learning approaches, e.g. k-nearest neighbor, support vector machines (SVM), semisupervised SVM etc. to detect false data injection attacks. Sedghi and Jonckheere [41] use a conditional covariance test to detect stealthy attacks. In their approach they show that the phase angle measurements form a Gaussian Markov random field and use the local Markov property of the phase angles to estimate the sample correlation matrix. Any attack in the data stream that significantly changes the structure of the estimated covariance matrix triggers the alarm.

1.3 Major Contributions

1.3.1 Performance Bounds for the Sensor Placement Problem

We consider a static discrete optimization problem where we have n discrete node locations where we can deploy sensors and we have m sensors to place ($m << n$). The objective is to minimize the sum of the mean squared estimation error at all node locations. This has some close ties to [1, 6] and has applications to optimal PMU placement as well as other energy problems. Examples include placement of meters such as Advanced Metering Infrastructure (AMI) on the distribution grid or deployment of environmental resource sensors where distributed PV solar panels are located.

The optimization criteria we consider is maximizing the trace of the inverse of the information matrix, known as the A- optimality design of experiments, which has also been considered in [1, 6, 7, 25]. A key difference between our work and the others is that here we consider a variety of computationally efficient approximation algorithms for the sensor placement problem and come up with analytical nested lower and upper bounds (depending on the structure of the correlation matrix) for the cost function of the optimal sensor placement. With a large number of suboptimal algorithms presented in the literature to solve the optimal

sensor placement problem, these analytical bounds provide a benchmark to compare the performance of these suboptimal algorithms in a large network, where the optimal solution is not available. In [6], the authors present a bound on the optimal PMU placement problem under the assumption that the reward function is submodular. Under the submodularity assumption, the optimal solution is upper bounded by the greedy solution factored by $e/(e-1)$. This bound is usually not very tight, because it does not take the covariance matrix structure (i.e., eigenvalues, eigenvectors etc.) into account. In contrast, we do not assume any submodularity condition and the upper bounds are obtained analytically in terms of the eigenvalues of the covariance matrix. The main results on these upper bounds have been published in author's works [43–46], in collaboration with A. Kuh, A. Kavcic, and T. Tanaka.

1.3.2 Online Outlier Detection

In this part we investigate two methods for online outlier detection that are able to deal with large data streams and suitable for bad data identification in the power grid.

The first algorithm we propose is an online Kernel Density Estimation (KDE) method for estimation of the probability distribution of the data. Since an outlier is a deviation from the normal behavior of the system it has a low probability of occurrence. We also pre-cluster the data using a Shared Nearest Neighbor (SNN) clustering algorithm to obtain a better estimate of the bandwidth matrix used in the KDE. The proposed KDE based outlier detection technique is a local algorithm that only requires the data from a bus and its neighbors to detect outliers at the bus. Thus, this requires only a one-hop communication system between the neighboring buses.

While the KDE based outlier detection algorithm works well when the number of neighbor buses is small, in a large power network the number of neighbors increases. This makes the KDE based outlier detection computationally expensive. To address the computational issues, next we propose an online outlier detection algorithm based on least squares one-class support vector machine (SVM) classifiers for detecting outliers in a large power grid network,

in a decentralized manner. The one-class (OC) SVM is an unsupervised learning method, proposed by Schölkopf et al. in [47] and further advocated by Tax and Duin in [48], to extract regions in the input space where most of the training objects lie. A least squares version of the one-class SVM was proposed by Choi in [49] such that the solution can be obtained by solving a linear system instead of a quadratic programming problem in the standard one-class SVM. However, this advantage comes at the cost of loss of sparsity of the support vectors (SVs). Several approaches to sparsification of kernel-based solutions have been proposed in the literature [50–53]. In [54] the authors obtain a sparse set of support vectors for least-squares one-class SVM classifier for detection of abnormal events in video surveillance. However, their approach still requires the storage of all the training objects to obtain the decision hyperplane. In this paper, we utilize the approximate linear dependence (ALD) criterion [51] to obtain a sparse representation of the decision hyperplane in least-squares one-class SVM. Our approach has lower computational complexity and memory requirement than the non-sparse least-squares one-class SVM while still maintaining similar performance. Also, the parameters are updated recursively, thus making this method suitable for application on a data stream. The algorithm can be used as a distributed algorithm to detect outliers before the data is sent to a central processing terminal. Since the proposed method does not depend on the output of the state estimator, it saves valuable time in providing real-time information about the data quality. The main results on online outlier detection in the power grid have been published in the author's works [55–57], in collaboration with A. Kuh, Y. Weng, and M. Ilić.

1.4 Thesis Outline

This dissertation investigates the performance of the sensor placement algorithms and the detection of outliers in the sensor data. Chapter 2 gives a formulation of the sensor placement problem along with some proposed suboptimal algorithms. A family of nested bounds to the optimal solution are the presented to evaluate the performance of the suboptimal al-

gorithms. The performances of these proposed bounds are then numerically evaluated using simulations.

In Chapter 3, we propose the online Kernel Density Estimations based outlier detection algorithm for bad data detection in SCADA data. Spatial and temporal models of outlier detection are proposed and evaluated using numerical simulations. Chapter 4 proposes a sparse online least-squares one-class SVM based algorithm for outlier detection in a distributed manner. The detection rates of the algorithm are evaluated for gross measurement errors and false data injection attacks in the SCADA data. Finally, Chapter 5 wraps up the thesis and discusses some possible future research directions.

Notations: Upper case and lower case letters denote random variables and their realizations, respectively; underlined letters stand for vectors; boldface upper case letters denote matrices, and $\mathbf{I}$ denotes the identity matrix; $\langle \cdot\,, \cdot \rangle$ denotes a matrix pencil; $(\cdot)^T$ and $\mathbf{E}(\cdot)$ stand for transposition and expectation, respectively; $|\cdot|$ denotes the cardinality, absolute value and matrix determinant for sets, scalars and matrices, respectively; $\|\cdot\|$ denotes the L_2 norm of vectors.

2

Performance Bounds for the Sensor Placement Algorithms

2.1 Measurement Model

Conventional state estimators rely on the redundant measurements captured by supervisory control and data acquisition (SCADA) systems [2], which can only take non-synchronized measurements. The PMUs, on the other hand, can directly measure the states at the PMU-installed buses, and the states of all the connected buses (if enough channels are available). In fact, given the high measurement precision and reliability of the PMUs, we can consider

the PMU measurements to be low-noise refinements of certain states (exactly those states that are measured by the PMUs) [3]. We consider voltage magnitudes and phase angles as state variables that are initially estimated using nonlinear state estimator from SCADA data, and then further refined using sparse PMU measurements. We further explain this scenario below.

Assume there are n_b buses. Let V_k and Θ_k denote the voltage magnitude and angle of the kth bus, $k = 1, \ldots, n_b$. Let $\underline{V} = [V_1, V_2, \ldots, V_{n_b}]^T$ be the state vector representing the bus voltage magnitudes and $\underline{\Theta} = [\Theta_1, \Theta_2, \ldots, \Theta_{n_b}]^T$ be the state vector representing the corresponding phase angles. To make the state estimation more efficient in terms of storage and computational costs, we assume the voltage magnitudes and phases to be statistically independent random vectors [2, 9]. We further assume that all the PMUs are identical and take statistically independent voltage magnitude and phase measurements with variances σ_v^2 and σ_θ^2, respectively.

Let m_v and m_θ be the number of PMU voltage magnitude measurements and the number of PMU phase angle measurements, respectively, where $m_v \leq n_b$ and $m_\theta \leq n_b$. Let $\underline{Y}_v \in \mathbb{R}^{m_v}$ and $\underline{Y}_\theta \in \mathbb{R}^{m_\theta}$ be the PMU voltage magnitude measurement vector and PMU phase angle measurement vector, respectively. Then the PMU measurement model is:

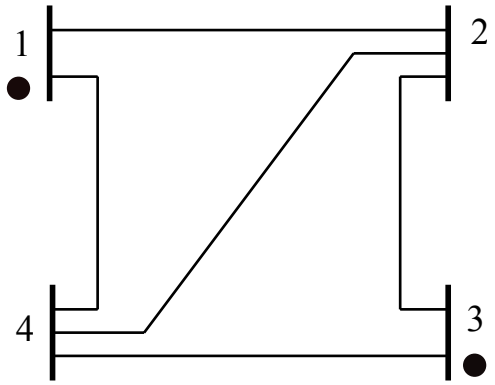
$$\underline{Y}_v = \mathbf{C}_v(\underline{V} + \sigma_v \underline{N}_v), \tag{2.1}$$

$$\underline{Y}_\theta = \mathbf{C}_\theta(\underline{\Theta} + \sigma_\theta \underline{N}_\theta), \tag{2.2}$$

where $\underline{N}_v$ and $\underline{N}_\theta$ are random noise vectors with mean zero and covariance matrix $\mathbf{I}_{n_b}$. We assume that the random noise vectors $\underline{N}_v$ and $\underline{N}_\theta$ are statistically independent of the state vectors. $\mathbf{C}_v$ and $\mathbf{C}_\theta$ are matrices that represent the positions of the PMU placements (see Example 2.1). These are binary matrices with orthonormal rows, where each row has one '1'. The positions of ones in the matrix $\mathbf{C}_v$ and $\mathbf{C}_\theta$ denote the position of the sensors. In order to provide a reference point for the phase angle measurements, we assume that a PMU

is always placed at the swing bus [3]. [1]

Example 2.1. *Figure 2.1 shows a* 4*-bus system [4] and matrices* $\mathbf{C}_v$ *and* $\mathbf{C}_\theta$ *when two PMUs are placed on buses* 1 *and* 3*.*



$$\mathbf{C}_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{C}_\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.1: PMU placement in a 4 bus system

Equations (2.1) and (2.2) represent separate models for PMU voltage magnitude and PMU phase angle measurements. We can combine (2.1) and (2.2) into a single model equation as

$$\begin{bmatrix} \underline{Y}_v \\ \underline{Z}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{C}_v & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_\theta \end{bmatrix} \left(\begin{bmatrix} \underline{Y} \\ \underline{\Theta} \end{bmatrix} + \begin{bmatrix} \sigma_v \underline{N}_v \\ \sigma_\theta \underline{N}_\theta \end{bmatrix} \right). \tag{2.3}$$

■

Next, we argue that we can capture the natures of models (2.1)-(2.3) using a single state vector $\underline{X} \in \mathbb{R}^n$ and a single measurement vector $\underline{Z} \in \mathbb{R}^m$ $(m \leq n)$ as

$$\underline{Y} = \mathbf{C}(\underline{X} + \sigma \underline{N}), \tag{2.4}$$

where $\underline{N}$ and $\underline{X}$ are statistically independent zero-mean random vectors with covariance

1. In the power flow problem, the power injection P_i is specified for all generator buses except one. This one bus, called a slack bus or swing bus or reference bus, is left open to balance the active power injections by accounting for the line losses. It is conventional, but completely arbitrary, to choose bus 1 as the swing bus [58]. The swing bus is specified to have a fixed voltage magnitude and phase angle, e.g., $\theta_1 = 0$, thus providing a reference for the phase angle at all the remaining buses.

matrices $\mathbf{I}_n$ and $\mathbf{\Sigma}_{\underline{X}}$, respectively. The following two examples illustrate this concept.

Example 2.2. *Under the following transformations,* (2.1) *and* (2.4) *are equivalent.*

$$\begin{aligned}
\underline{X} &= \underline{V} - \boldsymbol{E}(\underline{V}), & \sigma &= \sigma_v, \\
\underline{Y} &= \underline{Y}_v - \boldsymbol{E}(\underline{Y}_v), & n &= n_b, \\
\underline{N} &= \underline{N}_v, & m &= m_v, \\
\mathbf{C} &= \mathbf{C}_v.
\end{aligned}$$

■

Example 2.3. *Under the following transformations,* (2.3) *and* (2.4) *are equivalent.*

$$\begin{aligned}
\underline{X} &= \begin{bmatrix} \underline{V} - \boldsymbol{E}(\underline{V}) \\ \frac{\sigma_v}{\sigma_\theta}\left(\underline{\Theta} - \boldsymbol{E}(\underline{\Theta})\right) \end{bmatrix}, & \sigma &= \sigma_v, \\
\underline{Y} &= \begin{bmatrix} \underline{Y}_v - \boldsymbol{E}(\underline{Y}_v) \\ \frac{\sigma_v}{\sigma_\theta}\left(\underline{Y}_\theta - \boldsymbol{E}(\underline{Y}_\theta)\right) \end{bmatrix}, & n &= 2n_b, \\
\underline{N} &= \begin{bmatrix} \underline{N}_v \\ \underline{N}_\theta \end{bmatrix}, & m &= m_v + m_\theta, \\
\mathbf{C} &= \begin{bmatrix} \mathbf{C}_v & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_\theta \end{bmatrix}.
\end{aligned}$$

■

We assume that the network is observable using conventional SCADA measurements and $\mathbf{\Sigma}_{\underline{X}}$ is the state covariance matrix (estimated using the traditional nonlinear SE approaches), $\underline{X}$ and $\underline{N}$ are statistically independent and $\mathbf{C}$ is composed of m rows of the $n \times n$ identity matrix $\mathbf{I}_n$. We further simplify the model by considering only one measurement for each PMU. This assumption helps confirm the performance of the model in the worst case [26].

2.2 Problem Statement

We wish to utilize the newly obtained low-noise PMU measurement $\underline{Y}$ to make a refined estimate $\hat{\underline{X}}(\underline{Y})$ of the entire state vector $\underline{X}$. To this end, we use the linear minimum mean squared error estimator is given by [59]

$$\hat{\underline{X}}(\underline{Y}) = \mathbf{E}\left(\underline{X}\,\underline{Y}^T\right)\mathbf{E}\left(\underline{Y}\,\underline{Y}^T\right)^{-1}\underline{Y}. \tag{2.5}$$

The error is defined as $\underline{\mathcal{E}} = \underline{X} - \hat{\underline{X}}(\underline{Y})$ and the error covariance matrix is given by [59]

$$\mathbf{E}\left(\underline{\mathcal{E}}\,\underline{\mathcal{E}}^T\right) = \mathbf{\Sigma}_{\underline{X}} - \mathbf{E}\left(\underline{X}\,\underline{Y}^T\right)\mathbf{E}\left(\underline{Y}\,\underline{Y}^T\right)^{-1}\mathbf{E}\left(\underline{Y}\,\underline{X}^T\right), \tag{2.6}$$

where

$$\mathbf{E}\left(\underline{X}\,\underline{Y}^T\right) = \mathbf{\Sigma}_{\underline{X}}\mathbf{C}^T, \tag{2.7}$$

and

$$\mathbf{E}\left(\underline{Y}\,\underline{Y}^T\right) = \mathbf{C}\mathbf{\Sigma}_{\underline{X}}\mathbf{C}^T + \sigma^2\mathbf{I}_m. \tag{2.8}$$

Our task is to find the matrix $\mathbf{C}$ that minimizes the total error $\operatorname{tr}\mathbf{E}\left(\underline{\mathcal{E}}\,\underline{\mathcal{E}}^T\right)$.

Definition. *Let $\mathcal{C}^{[m\times n]}$ denote the* **set** *of all $m \times n$ matrices composed of m rows of the identity matrix $\mathbf{I}_n$.* ■

The optimization problem is then given by

$$\mathbf{C}^* = \underset{\mathbf{C}\in\mathcal{C}^{[m\times n]}}{\arg\min}\operatorname{tr}\mathbf{E}\left(\underline{\mathcal{E}}\,\underline{\mathcal{E}}^T\right) = \underset{\mathbf{C}\in\mathcal{C}^{[m\times n]}}{\arg\min}\mathbf{E}\left(\underline{\mathcal{E}}^T\underline{\mathcal{E}}\right). \tag{2.9}$$

Since the first term in (2.6) (i.e., $\mathbf{\Sigma}_{\underline{X}}$) does not depend on the choice of matrix $\mathbf{C}$, we can restate the optimization problem as an equivalent maximization problem using the following definition.

Definition. *Let the* **efficacy** *of matrix* $\mathbf{C}$ *be defined as*

$$J(\mathbf{C}) \stackrel{\Theta}{=} \operatorname{tr}\left\{ \boldsymbol{E}\left(\underline{X}\,\underline{Y}^T\right) \boldsymbol{E}\left(\underline{Y}\,\underline{Y}^T\right)^{-1} \boldsymbol{E}\left(\underline{Y}\,\underline{X}^T\right) \right\} \tag{2.10}$$

$$= \operatorname{tr}\left\{ \left[\mathbf{C}(\boldsymbol{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I})\mathbf{C}^T\right]^{-1} \mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}^2\mathbf{C}^T \right\}. \tag{2.11}$$

where the final equality follows from the properties of the trace operator. [Note that (2.11) has the form of the generalized Rayleigh quotient.] ■

The optimization problem (2.9) is then equivalent to

$$\mathbf{C}^* = \underset{\mathbf{C}\in\mathcal{C}^{[m\times n]}}{\arg\max}\, J(\mathbf{C}). \tag{2.12}$$

which is an integer programming problem of choosing m rows of the identity matrix $\mathbf{I}_n$ that maximize the efficacy. The optimum solutions to (2.12) requires an exhaustive search by testing all $\binom{n}{m}$ possible choices of m rows. Even for a moderately sized n and m this becomes computationally infeasible. In fact, the sensor placement problem is NP-complete [5].

We can further rewrite the efficacy to take advantage of the eigenstructure of the underlying matrices.

Definition. *Let* $\bar{\mathbf{C}}$ *denote* **a complement** *of* $\mathbf{C}$*, with constraints* $\bar{\mathbf{C}} \in \mathcal{C}^{[(n-m)\times n]}$ *and* $\bar{\mathbf{C}}\mathbf{C}^T = \mathbf{0}$*. [Note that* $\bar{\mathbf{C}}$ *may not be unique.]* ■

We perform the eigendecomposition of $\mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T = \mathbf{U}_{(\mathbf{C})}\mathbf{D}_{(\mathbf{C})}\mathbf{U}_{(\mathbf{C})}^T$ where the columns of $\mathbf{U}_{(\mathbf{C})}$ are the eigenvectors of $\mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T$ and $\mathbf{D}_{(\mathbf{C})}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T$. Let $\lambda_{(\mathbf{C}),1}, \cdots, \lambda_{(\mathbf{C}),m}$ be the eigenvalues of $\mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T$. Let the permutation matrix $\mathbf{P} = [\mathbf{C}^T, \bar{\mathbf{C}}^T]$ and note that

$$\mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}^2\mathbf{C}^T = \mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{P}\mathbf{P}^T\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T = \mathbf{U}_{(\mathbf{C})}\mathbf{D}_{(\mathbf{C})}^2\mathbf{U}_{(\mathbf{C})}^T + \mathbf{C}\boldsymbol{\Sigma}_{\underline{X}}\bar{\mathbf{C}}^T\bar{\mathbf{C}}\boldsymbol{\Sigma}_{\underline{X}}\mathbf{C}^T \tag{2.13}$$

Then combining this with (2.11) we have that

$$J(\mathbf{C}) = \operatorname{tr}\left\{\mathbf{U}_{(\mathbf{C})}\left[\mathbf{D}_{(\mathbf{C})} + \sigma^2\mathbf{I}\right]^{-1}\mathbf{D}^2_{(\mathbf{C})}\mathbf{U}^T_{(\mathbf{C})}\right\} + \operatorname{tr}\left\{\bar{\mathbf{C}}\underline{\Sigma}_{\underline{X}}\mathbf{C}^T\mathbf{U}_{(\mathbf{C})}\left[\mathbf{D}_{(\mathbf{C})} + \sigma^2\mathbf{I}\right]^{-1}\mathbf{U}^T_{(\mathbf{C})}\mathbf{C}\underline{\Sigma}_{\underline{X}}\bar{\mathbf{C}}^T\right\}. \tag{2.14}$$

The first term in (2.14) is the trace of a diagonal matrix and it contributes to the efficacy by summing the diagonal terms $\lambda^2_{(\mathbf{C}),i}/(\lambda_{(\mathbf{C}),i} + \sigma^2)$. The second term accounts for the state correlations. It is the second term that is most difficult to deal with when attempting to solve (2.12). Intuitively, we want to pick the matrix $\mathbf{C}$ such that the second term contributes considerably to the efficacy, i.e., we want to place sensors in locations that are highly correlated to the remaining states.

We can restate (2.14) using inner products (i.e., correlations).

Definition. *Let* $\underline{g}_i$ *be the* **vector of inner products** *between the i-th eigenvector in* $\mathbf{U}_{(\mathbf{C})}$ *and the columns of* $\mathbf{C}\underline{\Sigma}_{\underline{X}}\bar{\mathbf{C}}^T$*, i.e.,*

$$\underline{g}_i = \left(\bar{\mathbf{C}}\underline{\Sigma}_{\underline{X}}\mathbf{C}^T\right)\left(\mathbf{U}_{(\mathbf{C})}\underline{e}_i^T\right), \tag{2.15}$$

where $\underline{e}_i$ *is the i-th unit row vector and* $\left(\mathbf{U}_{(\mathbf{C})}\underline{e}_i^T\right)$ *is the i-th eigenvector in* $\mathbf{U}_{(\mathbf{C})}$*.* ■

The efficacy in (2.14) now takes the form

$$J(\mathbf{C}) = \sum_{i=1}^{m}\frac{\lambda^2_{(\mathbf{C}),i}}{\lambda_{(\mathbf{C}),i} + \sigma^2} + \sum_{i=1}^{m}\frac{\underline{g}_i^T\underline{g}_i}{\lambda_{(\mathbf{C}),i} + \sigma^2}. \tag{2.16}$$

We can readily interpret the second term in (2.16) as the contribution of the *energy* in the correlations (between sensor readings and the remaining states) to the efficacy. Clearly, we would like to find a matrix $\mathbf{C}$ so that the eigenvalues are large and the measurements are maximally correlated to the remaining states.

Problem (2.12) is an integer programming problem of choosing m rows of the identity matrix $\mathbf{I}_n$ that maximize the efficacy in (2.16). This can be solved by an exhaustive search

requiring testing all $\binom{n}{m}$ possible choices of m rows. Even for a moderately sized n and m this becomes computationally infeasible. The sensor placement problem is in fact NP-complete [5]. Section 2.3 gives computationally feasible approximation algorithms to solve (2.12) and Section 2.4 gives upper bounds to maximum efficacy in (2.12).

2.3 Efficacy-based Approximate Solutions

Since the optimization in (2.12) is difficult to perform, we resort to approximate solutions. Each approximate solution is in fact an ad-hoc solution because the exact solution requires an exhaustive search. If $\mathbf{C}$ is an ad-hoc solution to (2.12), then it provides a lower bound on the optimal efficacy $J(\mathbf{C}^*)$, i.e., $J(\mathbf{C}) \leq J(\mathbf{C}^*)$. Therefore, the search for good (suboptimal) solutions to (2.12) is equivalent to constructing tight lower bounds on $J(\mathbf{C}^*)$. Here we consider approximate solutions to the optimization problem requiring a much lower search complexity than $\mathcal{O}\left(\binom{n}{m}\right)$.

2.3.1 Expedient solution

This is a trivial approximate solution to consider. Let $J(\underline{e}_k)$ be the efficacy of the k-th unit row vector, i.e., the efficacy of the sensor placed at the location of the k-th state variable when $m = 1$. Then using (2.11) we have

$$J(\underline{e}_k) = \sum_{i=1}^{n} \frac{\left(\underline{e}_k \, \mathbf{\Sigma}_{\underline{X}} \, \underline{e}_i^T\right)^2}{\underline{e}_k \, \mathbf{\Sigma}_{\underline{X}} \, \underline{e}_k^T + \sigma^2}. \tag{2.17}$$

We rank the vectors $\underline{e}_k$ in descending order of their efficacies $J(\underline{e}_k)$. For any arbitrary m, we pick the m highest ranked vectors $\underline{e}_k$ and stack them to be the rows of the approximate solution $\mathbf{C}_E$. Clearly we have $J(\mathbf{C}_E) \leq J(\mathbf{C}^*)$.

Since this algorithm requires sorting and picking m highest ranked vectors $\underline{e}_k$, it has search

complexity at most $\mathcal{O}(n \log n)$. Thus this algorithm finds an approximate solution very fast. Hence the solution obtained by this algorithm can be used as a good starting point of an iterative algorithm [60].

2.3.2 Greedy solution

A greedy algorithm obtains an approximate solution to (2.12) by making a sequence of choices [61]. At each step t, it assumes that t sensor locations are fixed, and makes a greedy choice where to place the $(t+1)$-st sensor. Let $\mathbf{C}_G$ denote the solution provided by the greedy algorithm. The algorithm can be described by the following [61].

Algorithm 2.1: GREEDY ALGORITHM

Initialize: Set iteration $t = 1$ and choose $\mathbf{C}_t = \underline{e}^*$ such that $\underline{e}^* = \arg\max_{\underline{e} \in \mathcal{C}^{[1\times n]}} J(\underline{e})$.

while $t < m$ **do**

Find $\underline{e}^* = \arg\max_{\underline{e} \in \mathcal{C}^{[1\times n]}:\, \mathbf{C}_t \underline{e}^T = \mathbf{0}} J\left(\begin{bmatrix} \mathbf{C}_t \\ \underline{e} \end{bmatrix}\right)$.

Set $\mathbf{C}_{t+1} = \begin{bmatrix} \mathbf{C}_t \\ \underline{e}^* \end{bmatrix}$.

Increment: $t \leftarrow t + 1$.

end

Set $\mathbf{C}_G = \mathbf{C}_t$.

Note that the greedy solution may not be optimal even for $m = 2$, but it has search complexity $\mathcal{O}(mn)$ which is much smaller than $\mathcal{O}\left(\binom{n}{m}\right)$ required to find the optimal solution $\mathbf{C}^*$.

2.3.3 n-path greedy solution

We propose the n-path greedy method to compute n candidate solutions where each candidate solution is attained by starting the greedy algorithm using each of the unit row vectors $\underline{e}_k$, where $k = 1, \ldots, n$. Let $\mathbf{C}_m^{(k)}$ denote the candidate solution when the greedy algorithm is initiated with vector $\underline{e}_k$. Finally, we choose the n-path greedy solution $\mathbf{C}_{nG}$ to be the best

of the n different candidate solutions.

$$\mathbf{C}_{nG} = \underset{\mathbf{C}\in\left\{\mathbf{C}_m^{(1)},\mathbf{C}_m^{(2)},\cdots,\mathbf{C}_m^{(n)}\right\}}{\arg\max} J(\mathbf{C}). \tag{2.18}$$

The n-path greedy algorithm runs in polynomial time. It has search complexity $\mathcal{O}(mn^2)$ which is larger than the $\mathcal{O}(mn)$ search complexity of the plain greedy algorithm in Section 2.3.2, but the n-path greedy algorithm performs better than the plain greedy algorithm, thus giving a tighter lower bound $J(\mathbf{C}_{nG})$ on the optimal efficacy $J(\mathbf{C}^*)$, i.e., $J(\mathbf{C}_G) \leq J(\mathbf{C}_{nG}) \leq J(\mathbf{C}^*)$. [2]

2.3.4 Backtraced n-path solution

We now propose a *backtraced* version of the n-path greedy algorithm to solve the optimization problem (2.12). This algorithm solves optimization problem (2.12) by dividing the problem into smaller subproblems, which is similar to the heuristics of the dynamic programming algorithm [61, 62]. However, the sensor placement problem does not have the optimal substructure property. Thus the backtraced n-path solution is not optimal in general. In this algorithm we use a *bottom-up* approach to rank the subproblems in terms of their problem sizes, smallest first. We save the intermediate solutions of the subproblems in a table and later use them to solve larger subproblems. For our optimization problem, we define a subproblem of size (number of sensors) t as finding the best candidate solution of size $t-1$ for a newly added sensor in a fixed location. Therefore, for any arbitrary number of sensors t, we have n subproblems of size t. Let $\mathbf{C}_t^{(j)}$ be the solution to a subproblem of size t, where $t \in \{1, 2, \cdots, m\}$ is the number of sensors and $j \in \{1, 2, \cdots, n\}$ is the index of the subproblem. In short, the goal of the backtraced algorithm is to append the best existing solution

2. Further improvement on the performance of the n-path greedy algorithm may be possible by considering a larger initial search space, such as $\mathcal{O}(n^k)$, so that the locations of the first k sensors are guaranteed to be optimal. However, this comes at the cost of increased search complexity of $\mathcal{O}(mn^{k+1})$. Usually, there is no or only marginal performance gain since the n-path greedy solution often performs close or identical to the optimal solution.

$\mathbf{C}_{t-1}^{(j)}$ to a fixed $\underline{e}_k$ and thus construct a solution for a subproblem of size t.

Let $\mathbf{C}_{BT}$ denote the approximate solution to (2.12) computed by the backtraced n-path algorithm, in terms of the solutions of the subproblems as

$$\mathbf{C}_{BT} = \underset{\mathbf{C}\in\left\{\mathbf{C}_m^{(1)},\mathbf{C}_m^{(2)},\cdots,\mathbf{C}_m^{(n)}\right\}}{\arg\max} J\left(\mathbf{C}\right), \tag{2.19}$$

where $\left\{\mathbf{C}_m^{(1)}, \mathbf{C}_m^{(2)}, \cdots, \mathbf{C}_m^{(n)}\right\}$ is the set of the solutions to the subproblems of size m. The following procedure implements the backtraced n-path algorithm.

Algorithm 2.2: BACKTRACED n-PATH ALGORITHM

Initialize: Set iteration $t = 1$ and initial matrices $\mathbf{C}_1^{(1)} = \underline{e}_1$, $\mathbf{C}_1^{(2)} = \underline{e}_2$, $\cdots$, $\mathbf{C}_1^{(n)} = \underline{e}_n$.
while $t < m$ **do**
  Set $k \leftarrow 1$.
  while $k < n$ **do**
    Find $j^* = \arg\max_{j:\mathbf{C}_t^{(j)}\underline{e}_k^T=\mathbf{0}} J\left(\begin{bmatrix}\mathbf{C}_t^{(j)}\\ \underline{e}_k\end{bmatrix}\right)$.
    $\mathbf{C}_{t+1}^{(k)} = \begin{bmatrix}\mathbf{C}_t^{(j^*)}\\ \underline{e}_k\end{bmatrix}$.
    $k \leftarrow k + 1$.
  end
  $t \leftarrow t + 1$.
end
Set $\mathbf{C}_{DP} = \arg\max_{\mathbf{C}\in\left\{\mathbf{C}_t^{(1)},\cdots,\mathbf{C}_t^{(n)}\right\}} J(\mathbf{C})$

The backtraced version has the same search complexity $\mathcal{O}(mn^2)$ as the n-path greedy algorithm, and performs better than the plain greedy approximation, i.e., $J(\mathbf{C}_G) \leq J(\mathbf{C}_{BT}) \leq J(\mathbf{C}^*)$. However, we cannot provide an a-priori comparison between $J(\mathbf{C}_{nG})$ and $J(\mathbf{C}_{BT})$ without explicitly computing both values.

2.4 Upper Bounds on the Optimal Efficacy

It is clear from the previous section that there are numerous ways of obtaining a lower bound on the optimal efficacy $J(\mathbf{C}^*)$. However, to evaluate the performance of these lower bounds we want to obtain a numerically computable upper bound for the difference $J(\mathbf{C}^*) - J(\mathbf{C})$. One way to achieve this goal is to find a numerically computable upper bound, say $\bar{J}$, on the optimal $J(\mathbf{C}^*)$ such that

$$J(\mathbf{C}^*) - J(\mathbf{C}) \leq \bar{J} - J(\mathbf{C}). \tag{2.20}$$

Hence, we devote this section to finding a family of upper bounds $\bar{J}_k$ on the optimal efficacy $J(\mathbf{C}^*)$ by relaxing conditions on $\mathbf{C}$. In Subsection 2.4.1 we present some definitions which describe the relaxation of the optimization constraints for problem (2.12). Next, Subsection 2.4.2 gives the canonic theorems used to calculate the family of upper bounds. Specifically, in Lemma B, we devise a method of calculating a family of upper bounds if the optimal solution is available for some $k \leq m$. Finally, in Subsection 2.4.3, we show that these bounds are nested and can be calculated in terms of the generalized eigenvalues of matrix pencils, using Theorem 2.1 and Lemma B.

2.4.1 Definitions

To develop a family of upper bounds on the optimal efficacy, we generalize the reward function (efficacy), and generalize the optimization problem and its constraints. Instead of considering two matrices $\mathbf{\Sigma}_{\underline{X}}^2$ and $\mathbf{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I}$, in this section we consider a general matrix pencil $\langle \mathbf{A}, \mathbf{B} \rangle$, where $\mathbf{A}$ and $\mathbf{B}$ do not necessarily equal $\mathbf{\Sigma}_{\underline{X}}^2$ and $\mathbf{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I}$, respectively. Next, instead of considering a matrix $\mathbf{C}$ whose entries take values in the set $\{0, 1\}$, in this section we consider a generalized matrix $\mathbf{F}$ whose entries take values in $\mathbb{R}$. Finally, we introduce a modified optimization problem (different from the one in Section 2.2) that leads to the upper bounds. The following definitions set the stage.

Definition. *For two $n \times n$ matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, we define the* **efficacy** *of a matrix* $\mathbf{C}$*, with respect to the matrix pencil* $\langle \mathbf{A}, \mathbf{B} \rangle$*, as*

$$J_{\langle \mathbf{A}, \mathbf{B} \rangle}(\mathbf{C}) \triangleq \operatorname{tr}\left\{ \left(\mathbf{C}\mathbf{B}\mathbf{C}^T \right)^{-1} \mathbf{C}\mathbf{A}\mathbf{C}^T \right\}, \tag{2.21}$$

[should the inverse $(\mathbf{C}\mathbf{B}\mathbf{C}^T)^{-1}$ *exist].* ■

Definition. *For* $m \leq n$*, let* $\mathcal{F}^{[m \times n]}$ *be the* **set** *of all* $m \times n$ *matrices with rank* m*.* ■

Definition. *We define* $\mathbf{F}^*_{\langle \mathbf{A}, \mathbf{B} \rangle}$ *to be the argument that solves the following optimization problem*

$$\mathbf{F}^*_{\langle \mathbf{A}, \mathbf{B} \rangle} \triangleq \underset{\mathbf{F} \in \mathcal{F}^{[m \times n]}}{\arg\max}\, J_{\langle \mathbf{A}, \mathbf{B} \rangle}(\mathbf{F}) \tag{2.22}$$

$$= \underset{\mathbf{F} \in \mathcal{F}^{[m \times n]}}{\arg\max} \operatorname{tr}\left\{ \left(\mathbf{F}\mathbf{B}\mathbf{F}^T \right)^{-1} \mathbf{F}\mathbf{A}\mathbf{F}^T \right\}. \tag{2.23}$$

■

Definition. *We define* $J^*_{\langle \mathbf{A}, \mathbf{B} \rangle}$ *as the solution to the optimization problem in (2.22).*

$$J^*_{\langle \mathbf{A}, \mathbf{B} \rangle} \triangleq \max_{\mathbf{F} \in \mathcal{F}^{[m \times n]}} J_{\langle \mathbf{A}, \mathbf{B} \rangle}(\mathbf{F}) = J_{\langle \mathbf{A}, \mathbf{B} \rangle}\left(\mathbf{F}^*_{\langle \mathbf{A}, \mathbf{B} \rangle} \right). \tag{2.24}$$

■

2.4.2 Canonic Theorem

If $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ symmetric matrices, there exist n generalized eigenvectors $\underline{v}_1, \underline{v}_2, \cdots, \underline{v}_n$, with corresponding generalized eigenvalues $\theta_1, \theta_2, \cdots, \theta_n$ such that $\mathbf{A}\underline{v}_j = \theta_j \mathbf{B}\underline{v}_j$. [Note: $\theta_1, \theta_2, \cdots, \theta_n$ need not be distinct.] We arrange the generalized eigenvalues as the diagonal

elements of a diagonal matrix $\mathbf{D}$,

$$\mathbf{D} \triangleq \begin{bmatrix} \theta_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \theta_n \end{bmatrix}, \tag{2.25}$$

and we arrange the generalized eigenvectors as the columns of a matrix $\mathbf{V}$,

$$\mathbf{V} \triangleq \begin{bmatrix} \underline{v}_1 & \cdots & \underline{v}_n \end{bmatrix}. \tag{2.26}$$

Lemma A. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are symmetric* $n \times n$ *matrices and* $\mathbf{B}$ *is positive definite, then*

$$\mathbf{V}^T \mathbf{B} \mathbf{V} = \mathbf{I}, \tag{2.27}$$

and

$$\mathbf{V}^T \mathbf{A} \mathbf{V} = \mathbf{D}. \tag{2.28}$$

Proof. See in [63]. ■

Theorem 2.1. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be symmetric and* $\mathbf{B}$ *be positive definite, and let* $\mathbf{D}$ *and* $\mathbf{V}$ *denote the generalized eigenvalue matrix and generalized eigenvector matrix as in (2.25) and (2.26), respectively. If the eigenvalues are ordered as* $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_n \geq 0$*, then*

$$J^*_{\langle \mathbf{A}, \mathbf{B} \rangle} = \operatorname{tr} \left\{ \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{D} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix}^T \right\} = \sum_{j=1}^{m} \delta_j, \tag{2.29}$$

and

$$\mathbf{F}^*_{\langle \mathbf{A}, \mathbf{B} \rangle} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{V}^T = \begin{bmatrix} \underline{v}_1 & \cdots & \underline{v}_m \end{bmatrix}^T. \tag{2.30}$$

[Note: The solution $\mathbf{F}^*_{\langle \mathbf{A}, \mathbf{B} \rangle}$ *in* (2.30) *is not unique.]*

Proof. [64] provides a proof for this theorem. Here we provide an alternative proof using Lemma A in Appendix A.1.

■

Remark 2.1. *If* $\mathbf{A} = \mathbf{\Sigma}_{\underline{X}}^2$ *and* $\mathbf{B} = \mathbf{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I}$*, and* $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ *are the eigenvalues of* $\mathbf{\Sigma}_{\underline{X}}$*, then the generalized eigenvalues of the pencil* $\left\langle \mathbf{\Sigma}_{\underline{X}}^2,\, \mathbf{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I} \right\rangle$ *are*

$$\theta_j = \frac{\lambda_j^2}{\lambda_j + \sigma^2}. \tag{2.31}$$

Thus, using Theorem 2.1 we can write

$$J^*_{\left\langle \mathbf{\Sigma}_{\underline{X}}^2,\, \mathbf{\Sigma}_{\underline{X}} + \sigma^2\mathbf{I} \right\rangle} = \sum_{j=1}^{m} \frac{\lambda_j^2}{\lambda_j + \sigma^2}. \tag{2.32}$$

■

Theorem 2.1 provides an upper bound for the optimal efficacy $J_{\langle \mathbf{A},\, \mathbf{B} \rangle}(\mathbf{C}^*)$ in terms of the generalized eigenvalues of the pencil $\langle \mathbf{A},\, \mathbf{B} \rangle$. We now devise a method to calculate a family of upper bounds for the optimal efficacy when the optimal solution is available for some $k \leq m$. These upper bounds get tighter as k increases. To develop a family of upper bounds, we find it useful to solve a series of modified efficacy maximization problems for all $k \leq m$. The next definition addresses the modified efficacy maximization problem.

Definition 2.1. *For any* $k \leq m$*, we define* $\mathbf{F}^{(k)*}_{\langle \mathbf{A},\, \mathbf{B} \rangle}$ *and* $J^{(k)*}_{\langle \mathbf{A},\, \mathbf{B} \rangle}$ *as the solution pair of the following modified efficacy maximization*

$$\mathbf{F}^{(k)*}_{\langle \mathbf{A},\, \mathbf{B} \rangle} \stackrel{\Delta}{=} \underset{\mathbf{F} \in \mathcal{F}^{[(m-k)\times(n-k)]}}{\arg\max}\; J_{\langle \mathbf{A},\, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right), \tag{2.33}$$

and

$$J^{(k)*}_{\langle \mathbf{A},\, \mathbf{B} \rangle} \stackrel{\Delta}{=} \max_{\mathbf{F} \in \mathcal{F}^{[(m-k)\times(n-k)]}} J_{\langle \mathbf{A},\, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) \tag{2.34}$$

$$= J_{\langle \mathbf{A},\, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^{(k)*}_{\langle \mathbf{A},\, \mathbf{B} \rangle} \end{bmatrix} \right). \tag{2.35}$$

■

In order to solve the modified efficacy maximization problem in (2.33) - (2.34), it is convenient to split the efficacy

$$J_{\langle \mathbf{A}, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right)$$

into two terms such that

1. the first term does not depend on $\mathbf{F}$, and
2. the second term equals the efficacy of $\mathbf{F}$ with respect to a modified pencil of lower dimensions.

We formulate the split in the following lemma.

Lemma B. *In the optimization problem (2.34), the efficacy can be expressed as*

$$J_{\langle \mathbf{A}, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) = t_k + J_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}(\mathbf{F}), \tag{2.36}$$

where the additive term t_k and the modified pencil $\langle \mathbf{A}_k, \mathbf{B}_k \rangle$ satisfy

$$t_k = \operatorname{tr} \left\{ \mathbf{A} \begin{bmatrix} \mathbf{P}_k^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right\} \tag{2.37}$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{P}_k^{-1}\mathbf{Q}_k \\ -\mathbf{I}_{n-k} \end{bmatrix}^T \mathbf{A} \begin{bmatrix} \mathbf{P}_k^{-1}\mathbf{Q}_k \\ -\mathbf{I}_{n-k} \end{bmatrix} \tag{2.38}$$

$$\mathbf{B}_k = \mathbf{R}_k - \mathbf{Q}_k^T \mathbf{P}_k^{-1} \mathbf{Q}_k \tag{2.39}$$

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix}^T \mathbf{B} \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix} \tag{2.40}$$

$$\mathbf{Q}_k = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix}^T \mathbf{B} \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{n-k} \end{bmatrix} \tag{2.41}$$

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{n-k} \end{bmatrix}^T \mathbf{B} \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{n-k} \end{bmatrix} \tag{2.42}$$

Proof. See Appendix A.2. ■

Lemma B now lets us express the solution of the the modified optimization problem (2.33)-(2.34) equivalently as the solution of a regular efficacy maximization (i.e., using Theorem 2.1), but for a modified matrix pencil. Hence we have the following corollary of Theorem 2.1.

Corollary 2.1.1.

$$\mathbf{F}^{(k)*}_{\langle \mathbf{A}, \mathbf{B} \rangle} = \mathbf{F}^{*}_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}, \tag{2.43}$$

and

$$J^{(k)*}_{\langle \mathbf{A}, \mathbf{B} \rangle} = t_k + J^{*}_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}. \tag{2.44}$$

Proof. In (2.36), t_k does not depend on $\mathbf{F}$. Therefore, (2.43) and (2.44) hold. ■

Remark 2.2.

$$J^{(0)*}_{\langle \mathbf{A}, \mathbf{B} \rangle} = J^{*}_{\langle \mathbf{A}, \mathbf{B} \rangle}. \tag{2.45}$$

■

Remark 2.3.

$$J^{(m)*}_{\langle \mathbf{A}, \mathbf{B} \rangle} = t_m. \tag{2.46}$$

■

From Corollary 2.1.1 we observe that the nested bounds are calculated in terms of the generalized eigenvalues of a matrix pencil with smaller dimensions.

2.4.3 Nested Bounds

We dedicate this section to applying the upper bounds, obtained for the modified optimization problem in Section 2.4.2, to the optimization problem defined in Section 2.2. We obtain these nested upper bounds assuming that the optimal solution to (2.12) is calculable for any $k \leq m$. We define the nested upper bounds $\bar{J}_k$ as follows.

Definition 2.2.

$$\bar{J}_k \triangleq \max_{\mathbf{C}\in\mathcal{C}^{[k\times n]}} \left\{ \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}\mathbf{C}^T=\mathbf{0}}} J_{\left\langle \mathbf{\Sigma}^2_{\underline{X}},\, \mathbf{\Sigma}_{\underline{X}}+\sigma^2\mathbf{I} \right\rangle} \left(\begin{bmatrix} \mathbf{C} \\ \mathbf{F} \end{bmatrix} \right) \right\}. \tag{2.47}$$

■

From Remark 2.2, we clearly see that $\bar{J}_0 \geq J(\mathbf{C}^*)$. We next show that $\bar{J}_k \geq J(\mathbf{C}^*)$ for any $k \leq m$.

Theorem 2.2. *For any $k \leq m$,*

$$\bar{J}_k \geq J\left(\mathbf{C}^*\right). \tag{2.48}$$

Proof.

$$J(\mathbf{C}^*) = \max_{\mathbf{C}\in\mathcal{C}^{[m\times n]}} J(\mathbf{C}) \tag{2.49}$$

$$= \max_{\mathbf{C}_1\in\mathcal{C}^{[k\times n]}} \; \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[(m-k)\times n]} \\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} \right) \tag{2.50}$$

$$\leq \max_{\mathbf{C}_1\in\mathcal{C}^{[k\times n]}} \; \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}\mathbf{C}_1^T=\mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C}_1 \\ \mathbf{F} \end{bmatrix} \right) \tag{2.51}$$

$$= \bar{J}_k, \tag{2.52}$$

where the inequality follows from the set relationship $\mathcal{C}^{[(m-k)\times n]} \subset \mathcal{F}^{[(m-k)\times n]}$. ■

Remark 2.4. *Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathbf{\Sigma}_{\underline{X}}$. Then, for $k = 0$,*

$$\bar{J}_0 = J^*_{\left\langle \mathbf{\Sigma}^2_{\underline{X}}, \mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \right\rangle} = \sum_{j=1}^{m} \frac{\lambda_j^2}{\lambda_j + \sigma^2}. \tag{2.53}$$

■

Remark 2.5. *When $k = m$,*

$$\bar{J}_m = J(\mathbf{C}^*) \tag{2.54}$$

■

We now show that the upper bounds are nested.

Theorem 2.3. *For any $k \leq m - 1$,*

$$\bar{J}_k \geq \bar{J}_{k+1}. \tag{2.55}$$

Proof.

$$\bar{J}_{k+1} = \max_{\mathbf{C} \in \mathcal{C}^{[(k+1)\times n]}} \max_{\substack{\mathbf{F} \in \mathcal{F}^{[(m-k-1)\times n]} \\ \mathbf{F}\mathbf{C}^T = \mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C} \\ \mathbf{F} \end{bmatrix} \right) \tag{2.56}$$

$$= \max_{\mathbf{C}_1 \in \mathcal{C}^{[k\times n]}} \max_{\substack{\underline{e} \in \mathcal{C}^{[1\times n]} \\ \mathbf{C}_1 \underline{e}^T = \mathbf{0}}} \max_{\substack{\mathbf{F} \in \mathcal{F}^{[(m-k-1)\times n]} \\ \mathbf{F}\mathbf{C}_1^T = \mathbf{0} \\ \mathbf{F}\underline{e}^T = \mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C}_1 \\ \underline{e} \\ \mathbf{F} \end{bmatrix} \right) \tag{2.57}$$

$$\leq \max_{\mathbf{C}_1 \in \mathcal{C}^{[k\times n]}} \max_{\substack{\underline{f} \in \mathcal{F}^{[1\times n]} \\ \mathbf{C}_1 \underline{f}^T = \mathbf{0}}} \max_{\substack{\mathbf{F} \in \mathcal{F}^{[(m-k-1)\times n]} \\ \mathbf{F}\mathbf{C}_1^T = \mathbf{0} \\ \mathbf{F}\underline{f}^T = \mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C}_1 \\ \underline{f} \\ \mathbf{F} \end{bmatrix} \right) \tag{2.58}$$

$$= \max_{\mathbf{C}_1 \in \mathcal{C}^{[k\times n]}} \max_{\substack{\mathbf{F}_1 \in \mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}_1 \mathbf{C}_1^T = \mathbf{0}}} J\left(\begin{bmatrix} \mathbf{C}_1 \\ \mathbf{F}_1 \end{bmatrix} \right) \tag{2.59}$$

$$= \bar{J}_k, \tag{2.60}$$

where the inequality is the consequence of the relationship $\mathcal{C}^{[1\times n]} \subset \mathcal{F}^{[1\times n]}$. ■

Corollary 2.3.1.

$$\sum_{j=1}^{m} \frac{\lambda_j^2}{\lambda_j + \sigma^2} = \bar{J}_0 \geq \bar{J}_1 \geq \cdots \geq \bar{J}_m = J(\mathbf{C}^*). \tag{2.61}$$

Proof. Combine (2.53) - (2.55). ■

We next want to utilize Theorem 2.1 (more specifically, Corollary 2.1.1) to efficiently compute the upper bounds $\bar{J}_k$. To that end, we define the matrix pencil $\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})} \right\rangle$ as a permutation of the matrix pencil $\left\langle \mathbf{\Sigma}_{\underline{X}}^2, \mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \right\rangle$.

Definition 2.3. *Let* $\bar{\mathbf{C}}$ *denote* **a complement** *of* $\mathbf{C}$*, with constraints* $\bar{\mathbf{C}} \in \mathcal{C}^{[(n-m)\times n]}$ *and* $\bar{\mathbf{C}}\mathbf{C}^T = \mathbf{0}$. ■

Definition 2.4. *We define*

$$\mathbf{A}_{(\mathbf{C})} = \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix} \mathbf{\Sigma}_{\underline{X}}^2 \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix}^T, \tag{2.62}$$

and

$$\mathbf{B}_{(\mathbf{C})} = \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix} \left(\mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \right) \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix}^T. \tag{2.63}$$

■

Combining Theorem 2.1 and Lemma B, we now reformulate the upper bounds $\bar{J}_k$ so that the bounds can be calculated in terms of the generalized eigenvalues of a matrix pencil of smaller dimensions.

Corollary 2.3.2.

$$\bar{J}_k = \max_{\mathbf{C} \in \mathcal{C}^{[k\times n]}} J^{(k)*}_{\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})} \right\rangle}. \tag{2.64}$$

Proof. Let $\mathbf{F} \in \mathcal{F}^{[(m-k)\times(n-k)]}$ and let $\mathbf{F}_1 = \mathbf{F}\bar{\mathbf{C}}$, for any $\mathbf{C} \in \mathcal{C}^{[k\times n]}$. Then, from (2.21), (2.62) and (2.63), it follows that

$$J_{\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})} \right\rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) = J_{\left\langle \mathbf{\Sigma}_{\underline{X}}^2, \mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \right\rangle} \left(\begin{bmatrix} \mathbf{C} \\ \mathbf{F}_1 \end{bmatrix} \right). \tag{2.65}$$

Since $\text{rank}(\mathbf{F}_1) = \text{rank}(\mathbf{F}) = m - k$, and $\mathbf{F}_1\mathbf{C}^T = \mathbf{0}$, using (2.34), (2.47) and (2.65) we can write

$$\bar{J}_k = \max_{\mathbf{C}\in\mathcal{C}^{[k\times n]}} \max_{\substack{\mathbf{F}_1\in\mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}_1\mathbf{C}^T=\mathbf{0}}} J_{\left\langle \mathbf{\Sigma}^2_{\underline{X}}, \mathbf{\Sigma}_{\underline{X}}+\sigma^2\mathbf{I}\right\rangle}\left(\begin{bmatrix}\mathbf{C}\\ \mathbf{F}_1\end{bmatrix}\right) \tag{2.66}$$

$$= \max_{\mathbf{C}\in\mathcal{C}^{[k\times n]}} \max_{\mathbf{F}\in\mathcal{F}^{[(m-k)\times(n-k)]}} J_{\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})}\right\rangle}\left(\begin{bmatrix}\mathbf{I}_k & \mathbf{0}\\ \mathbf{0} & \mathbf{F}\end{bmatrix}\right) \tag{2.67}$$

$$= \max_{\mathbf{C}\in\mathcal{C}^{[k\times n]}} J^{(k)*}_{\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})}\right\rangle}. \tag{2.68}$$

■

For any $k \leq m$, the computation of the upper bound $\bar{J}_k$ requires searching over all $\binom{n}{k}$ matrices $\mathbf{C} \in \mathcal{C}^{[k\times n]}$. Therefore, computation of $\bar{J}_k$ has search complexity $\mathcal{O}\left(\binom{n}{k}\right)$.

2.4.3.1 Nested Bounds with Constraints

So far we have considered the sensor placement problem without any constraints on the possible sensor locations. However, in many applications there are constraints on certain sensor location, e.g., topological constraints, zero-injection buses in the power grid etc. In this section, we develop a family of upper bounds when

1. some $k \leq m$ optimal sensor locations are available and,
2. some $\ell \leq n - m$ optimal locations are available where the sensors cannot be placed.

In fact, either of the two smaller optimizations can be replaced by imposing constraints on the sensor network. The nested upper bounds $\bar{H}_{k,\ell}$ are defined as follows

Definition. *For any $k < m$ and $\ell < n - m$,*

$$\bar{H}_{k,\ell} \stackrel{\Delta}{=} \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}\in\mathcal{C}^{[k\times n]} \\ \mathbf{C}\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}[\mathbf{C}^T\ \mathbf{C}_1^T]=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}\\ \mathbf{F}\end{bmatrix}\right) \tag{2.69}$$

■

Theorem 2.4.

$$\bar{H}_{k,\ell} \geq J\left(\mathbf{C}^*\right) \tag{2.70}$$

Proof.

$$\begin{aligned} J\left(\mathbf{C}^*\right) &= \max_{\mathbf{C}\in\mathcal{C}^{[m\times n]}} J\left(\mathbf{C}\right) \\ &= \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[k\times n]} \\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{C}_3\in\mathcal{C}^{[(m-k)\times n]} \\ \mathbf{C}_3[\mathbf{C}_2^T\ \mathbf{C}_1^T]=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}_2\\ \mathbf{C}_3\end{bmatrix}\right) \\ &\leq \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[k\times n]} \\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k)\times n]} \\ \mathbf{F}[\mathbf{C}_2^T\ \mathbf{C}_1^T]=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}_2\\ \mathbf{F}\end{bmatrix}\right) = \bar{H}_{k,\ell}, \end{aligned}$$

where the inequality follows from $\mathcal{C}^{[(m-k)\times n]} \subset \mathcal{F}^{[(m-k)\times n]}$. ■

Remark 2.6. $\bar{H}_{0,0} = J(\mathbf{F}^*)$.

Remark 2.7. $\bar{H}_{m,\ell} = \bar{H}_{k,n-m} = J(\mathbf{C}^*)$

Remark 2.8. *For any $k \leq m$,*

$$\bar{H}_{k,0} = \bar{J}_k. \tag{2.71}$$

Like the family of the upper bounds $\bar{J}_k$, the upper bounds $\bar{H}_{k,\ell}$ are also nested.

Theorem 2.5. *For any $k \leq m-1$ and $\ell \leq n-m$,*

$$\bar{H}_{k,\ell} \geq \bar{H}_{k+1,\ell}\ , \tag{2.72}$$

and for any $k \leq m$ and $\ell \leq n-m-1$,

$$\bar{H}_{k,\ell} \geq \bar{H}_{k,\ell+1}. \tag{2.73}$$

Proof. From (2.69), we have

$$\begin{aligned}
\bar{H}_{k+1,\ell} &= \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}\in\mathcal{C}^{[(k+1)\times n]}\\ \mathbf{C}\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k-1)\times n]}\\ \mathbf{F}\mathbf{C}_1^T=\mathbf{0}\\ \mathbf{F}\mathbf{C}^T=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}\\ \mathbf{F}\end{bmatrix}\right)\\
&= \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[k\times n]}\\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\underline{e}\in\mathcal{C}^{[1\times n]}\\ \underline{e}\mathbf{C}_2^T=\mathbf{0}\\ \underline{e}\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k-1)\times n]}\\ \mathbf{F}[\mathbf{C}_2^T\ \underline{e}^T]=\mathbf{0}\\ \mathbf{F}\mathbf{C}_1^T=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}_2\\ \underline{e}\\ \mathbf{F}\end{bmatrix}\right)\\
&\le \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[k\times n]}\\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\underline{f}\in\mathcal{F}^{[1\times n]}\\ \underline{f}\mathbf{C}_2^T=\mathbf{0}\\ \underline{f}\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}\in\mathcal{F}^{[(m-k-1)\times n]}\\ \mathbf{F}[\mathbf{C}_2^T\ \underline{f}^T]=\mathbf{0}\\ \mathbf{F}\mathbf{C}_1^T=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}_2\\ \underline{f}\\ \mathbf{F}\end{bmatrix}\right)\\
&= \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\substack{\mathbf{C}_2\in\mathcal{C}^{[k\times n]}\\ \mathbf{C}_2\mathbf{C}_1^T=\mathbf{0}}} \max_{\substack{\mathbf{F}_1\in\mathcal{F}^{[(m-k)\times n]}\\ \mathbf{F}_1\mathbf{C}_2^T=\mathbf{0}\\ \mathbf{F}_1\mathbf{C}_1^T=\mathbf{0}}} J\left(\begin{bmatrix}\mathbf{C}_2\\ \mathbf{F}_1\end{bmatrix}\right) = \bar{H}_{k,\ell},
\end{aligned}$$

where the inequality follows from $\mathcal{C}^{[1\times n]} \subset \mathcal{F}^{[1\times n]}$. Equation (2.73) can be proved similarly. ■

From Remark 2.8, we see that the upper bounds $\bar{H}_{k,\ell}$ are related to $\bar{J}_k$ which is computed by maximizing $J^{(k)*}_{\left\langle \mathbf{A}_{(\mathbf{C})}, \mathbf{B}_{(\mathbf{C})} \right\rangle}$ (Corollary 2.3.2). We can utilize this relationship to efficiently compute the family of upper bounds $\bar{H}_{k,\ell}$ as follows.

Corollary 2.5.1.

$$\bar{H}_{k,\ell} = \max_{\mathbf{C}_1\in\mathcal{C}^{[\ell\times n]}} \max_{\mathbf{C}\in\mathcal{C}^{[k\times(n-\ell)]}} J^{(k)*}_{\left\langle \tilde{\mathbf{A}}, \tilde{\mathbf{B}} \right\rangle} \tag{2.74}$$

where the modified pencil $\left\langle \tilde{\mathbf{A}}, \tilde{\mathbf{B}} \right\rangle$ *is computed by*

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix} \bar{\mathbf{C}}_1 \mathbf{\Sigma}_{\underline{X}}^2 \bar{\mathbf{C}}_1^T \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix}^T \tag{2.75}$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix} \bar{\mathbf{C}}_1 (\mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I}) \bar{\mathbf{C}}_1^T \begin{bmatrix} \mathbf{C} \\ \bar{\mathbf{C}} \end{bmatrix}^T \tag{2.76}$$

Proof. From (2.69) we have

$$\bar{H}_{k,\ell} = \max_{\mathbf{C}_1 \in \mathcal{C}^{[\ell \times n]}} \max_{\substack{\mathbf{C}_2 \in \mathcal{C}^{[k \times n]} \\ \mathbf{C}_2 \mathbf{C}_1^T = \mathbf{0}}} \max_{\substack{\mathbf{F}_1 \in \mathcal{F}^{[(m-k) \times n]} \\ \mathbf{F}_1 [\mathbf{C}_2^T \ \mathbf{C}_1^T] = \mathbf{0}}} J_{\langle \mathbf{\Sigma}_{\underline{X}}^2, \mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \rangle} \left(\begin{bmatrix} \mathbf{C}_2 \\ \mathbf{F}_1 \end{bmatrix} \right) \tag{2.77}$$

Since $\mathbf{C}_2 \mathbf{C}_1^T = \mathbf{0}$ and $\mathbf{F}_1 \mathbf{C}_1^T = \mathbf{0}$, we can write $\mathbf{C}_2$ and $\mathbf{F}_1$ as linear transformations of $\bar{\mathbf{C}}_1$ such that

$$\begin{aligned} \mathbf{C}_2 &= \mathbf{C} \bar{\mathbf{C}}_1 \quad \text{where } \mathbf{C} \in \mathcal{C}^{[k \times (n-\ell)]} \\ \mathbf{F}_1 &= \mathbf{F}_2 \bar{\mathbf{C}}_1 \quad \text{where } \mathbf{F}_2 \in \mathcal{F}^{[(m-k) \times (n-\ell)]}. \end{aligned}$$

Then from (2.21), we have,

$$J_{\langle \mathbf{\Sigma}_{\underline{X}}^2, \mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I} \rangle} \left(\begin{bmatrix} \mathbf{C}_2 \\ \mathbf{F}_1 \end{bmatrix} \right) = J_{\langle \bar{\mathbf{C}}_1 \mathbf{\Sigma}_{\underline{X}}^2 \bar{\mathbf{C}}_1^T, \bar{\mathbf{C}}_1 (\mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I}) \bar{\mathbf{C}}_1^T \rangle} \left(\begin{bmatrix} \mathbf{C} \\ \mathbf{F}_2 \end{bmatrix} \right) \tag{2.78}$$

Because $\text{rank}(\mathbf{F}_1) = \text{rank}(\mathbf{F}_2) = m - k$ and $\mathbf{F}_1 \mathbf{C}_2^T = \mathbf{0}$, we have $\mathbf{F}_2 \mathbf{C}^T = \mathbf{0}$. Let $\mathbf{F} \in \mathcal{F}^{[(m-k) \times (n-k-\ell)]}$ and let $\mathbf{F}_2 = \mathbf{F} \bar{\mathbf{C}}$. Then from (2.21), (2.75) and (2.76) it follows that

$$J_{\langle \tilde{\mathbf{A}}, \tilde{\mathbf{B}} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) = J_{\langle \bar{\mathbf{C}}_1 \mathbf{\Sigma}_{\underline{X}}^2 \bar{\mathbf{C}}_1^T, \bar{\mathbf{C}}_1 (\mathbf{\Sigma}_{\underline{X}} + \sigma^2 \mathbf{I}) \bar{\mathbf{C}}_1^T \rangle} \left(\begin{bmatrix} \mathbf{C} \\ \mathbf{F}_2 \end{bmatrix} \right) \tag{2.79}$$

Then combining (2.34), (2.77)-(2.79), we have

$$\bar{H}_{k,\ell} = \max_{\mathbf{C}_1 \in \mathcal{C}^{[\ell \times n]}} \max_{\mathbf{C} \in \mathcal{C}^{[k \times (n-\ell)]}} \max_{\mathbf{F} \in \mathcal{F}^{[(m-k) \times (n-k-\ell)]}} J_{\langle \tilde{\mathbf{A}}, \tilde{\mathbf{B}} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) \tag{2.80}$$

$$= \max_{\mathbf{C}_1 \in \mathcal{C}^{[\ell \times n]}} \max_{\mathbf{C} \in \mathcal{C}^{[k \times (n-\ell)]}} J^{(k)*}_{\langle \tilde{\mathbf{A}}, \tilde{\mathbf{B}} \rangle} \tag{2.81}$$

■

2.5 Projection-based Approximate Solutions

In the previous section, we obtained the subspace that maximizes the modified optimization problem (2.23) under the constraint $\mathbf{F} \in \mathcal{F}^{[m \times n]}$. The rows of the solution $\mathbf{F}^*$ i.e., $\underline{v}_1^T, \cdots, \underline{v}_m^T$ form a (not necessarily orthogonal) basis of the maximizing subspace. However, the solution subspace of the original optimization problem (2.21) has unit row vectors as its basis. Hence we may adopt the approximate solution strategy that finds the best subspace spanned by unit vectors using a projection approach. That is, we project n unit row vectors [3] onto the subspace $\mathbf{F}^*$ to find good choices for m rows of $\mathbf{C}$ (sensor locations).

So, instead of utilizing the efficacy as our reward function, here we utilize the norm of the projection as the reward function. Thereby, similar to the approaches taken in Section 2.3, we may now develop several projection-based approximate solutions (the expedient, greedy, and n-path greedy) that have varying search complexities.

3. The projection of a unit row vector $\underline{e}_k$ on the subspace $\mathbf{F}$ is given by $\mathbf{F}^T \left(\mathbf{F}\mathbf{F}^T\right)^{-1} \mathbf{F}\underline{e}_k^T$.

2.5.1 Expedient projection-based solution

This is a single shot solution. Let $z(\underline{e}_k)$ be the norm of the projection of the vector $\underline{e}_k$ onto the subspace spanned by $\mathbf{F}^*$

$$z(\underline{e}_k) \triangleq \left\| \mathbf{F}^{*T} \left(\mathbf{F}^* \mathbf{F}^{*T} \right)^{-1} \mathbf{F}^* \underline{e}_k^T \right\|. \tag{2.82}$$

We rank the vectors $\underline{e}_k$ in descending order of $z(\underline{e}_k)$. For any $m \leq n$, the approximate solution $\mathbf{C}_{E-proj}$ is constructed by picking the m highest ranked vectors $\underline{e}_k$ as rows of $\mathbf{C}_{E-proj}$. The projection based expedient solution has the same search complexity as the efficacy-based expedient solution, i.e., $\mathcal{O}(n \log n)$. Due to their low search complexity, both are extremely good candidates for a solution when a quick first-order approximation is needed, e.g., such when starting an iterative refinement algorithm [60]. A direct comparison of the efficacy-based expedient solution $\mathbf{C}_E$ and the projection-based expedient solution $\mathbf{C}_{E-proj}$ cannot be made *a priori*, but must be done on a case by case basis.

2.5.2 Greedy projection-based solution

From Corollary 2.3.2 and Lemma B, we know that for any lower order $k < m$, an optimal low order matrix $\mathbf{F}$ (achieving the upper bound $\bar{J}_k$) consists of the generalized eigenvectors of a lower-dimension pencil $\langle \mathbf{A}_k, \mathbf{B}_k \rangle$ given by (2.38) and (2.39). Assuming that the generalized eigenvalues of $\langle \mathbf{A}_k, \mathbf{B}_k \rangle$ are sorted in descending order, the matrix that upper bounds the efficacy is given by (2.30) as $\mathbf{F}^*_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle} = \begin{bmatrix} \mathbf{I}_{m-k+1} & \mathbf{0} \end{bmatrix} \mathbf{V}^T_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}$, where $\mathbf{V}^T_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}$ is the generalized eigenvector matrix of $\langle \mathbf{A}_k, \mathbf{B}_k \rangle$. The greedy algorithm takes advantage of this property. In each iteration $k < m$, we assume that the solution is known for k sensors, and we pick the $(k+1)$-st sensor location as the unit row vector $\underline{e}$ that has the highest norm when projected onto the subspace spanned by $\mathbf{F}^*_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}$.

The projection-based greedy algorithm has search complexity $\mathcal{O}(mn)$.

Algorithm 2.3: GREEDY PROJECTION-BASED ALGORITHM

Initialize: Set iteration $k = 1$, $\mathbf{C}_{k-1} = [\,]$, and calculate $\mathbf{A}_{(\mathbf{C}_{k-1})}$ and $\mathbf{B}_{(\mathbf{C}_{k-1})}$ using (2.62) and (2.63), respectively.

while $k < m$ **do**

Find $\mathbf{A}_k$, $\mathbf{B}_k$ using (2.38) and (2.39).

Set $\mathbf{F} = \begin{bmatrix}\mathbf{I}_{m-k+1} & \mathbf{0}\end{bmatrix} \mathbf{V}^T_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}$.

Find $\underline{e}^* = \arg\max_{\underline{e} \in \mathcal{C}^{[1\times(n-k+1)]}} \left\| \mathbf{F}^T \left(\mathbf{F}\mathbf{F}^T\right)^{-1} \mathbf{F}\underline{e}^T \right\|$.

Set $\mathbf{C}_k = \begin{bmatrix} \mathbf{C}_{k-1} \\ \underline{e}^* \bar{\mathbf{C}}^{k-1} \end{bmatrix}$.

Set $k \leftarrow k + 1$.

end

set $\mathbf{C}_{G-proj} = \mathbf{C}_k$

2.5.3 n-path greedy projection-based solution

In the projection-based n-path greedy algorithm, n candidate solutions are constructed by initializing the projection-based greedy algorithm with each of n unit row vectors $\underline{e}_k$. The best of the n candidate solutions is picked as the approximate solution. The projection-based n-path greedy algorithm runs in polynomial time with search complexity $\mathcal{O}(mn^2)$.

2.6 Simulation Results

We considered different test scenarios to evaluate the performances of the approximate solutions and the nested family of upper bounds $\bar{J}_k$ (for $k = 0$ to 5). In Subsection A we considered realizations of the covariance matrix $\mathbf{\Sigma}_{\underline{X}}$ generated at random for different system sizes $n = 20,$ and 50. Next in Subsection B we created a 5 by 5 grid with unit distance between neighboring points, i.e., each point is located at unit distance from its horizontal and vertical neighbors. We generated a covariance matrix using a Gaussian distribution, where the variance between the points, considered as vectors $\underline{x}_1$ and $\underline{x}_2$ is given by [1]:

$$\Sigma(\underline{x}_1, \underline{x}_2) = \exp\left(-\beta \left\|\underline{x}_1 - \underline{x}_2\right\|_2^2 \Big/ (2\pi)\right) \tag{2.83}$$

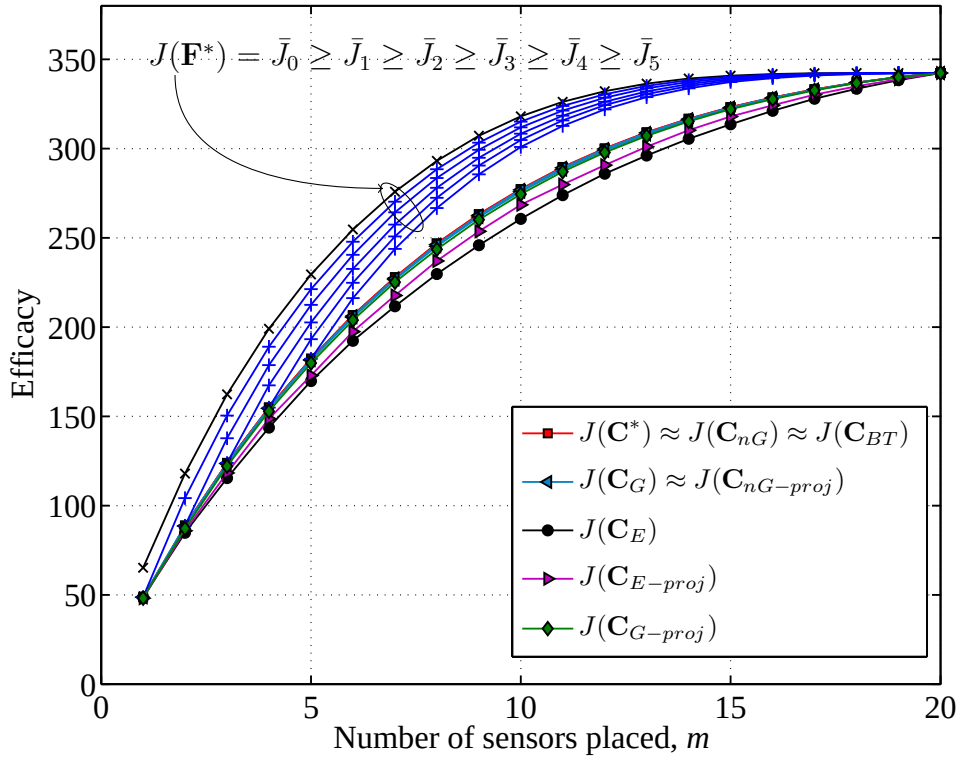


Figure 2.2: $n = 20$: average efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$ compared to the average optimal efficacy $J(\mathbf{C}^*)$

where $\beta > 0$. Finally we used the standard IEEE 57 bus test system [65] in Subsection C to show the performances of our proposed algorithms.

2.6.1 Simulations with data generated at random

First we consider the case where $n = 20$. We generated 100 realizations of the correlation matrix $\mathbf{\Sigma}_{\underline{X}}$ at random; σ^2 was kept constant for all realizations of $\mathbf{\Sigma}_{\underline{X}}$. We compared the efficacies of the expedient, optimal, greedy, n-path greedy, and backtraced n-path solutions and the upper bounds for each of the 100 realizations.

Fig. 2.2 shows the average efficacies and the upper bounds, averaged over 100 realizations. From Fig. 2.2 we note that both efficacy-based expedient solution and projection-based expedient solution get reasonably close to the optimal solution for each value of m. In this

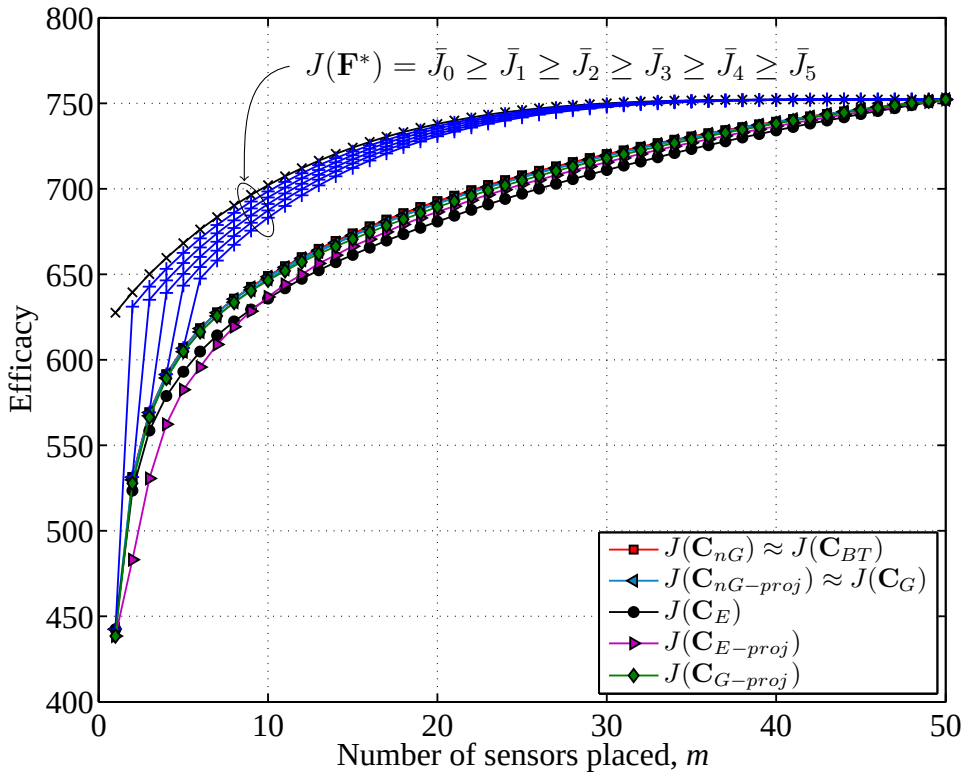


Figure 2.3: $n = 50$: average efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$

scenario, the projection-based expedient solution performs better than the efficacy-based one. However, the efficacy-based greedy solution performs better than the projection-based greedy solution. The projection-based n-path greedy does better than the efficacy-based greedy, but does worse than the efficacy-based n-Path greedy and backtraced n-path solution. Also the upper bound $J(\mathbf{F}^*) = \bar{J}_0$ is not tight, but the nested bounds $\bar{J}_k$ become tighter as k increases.

We then considered the case for $n = 50$. For this case we could not compute the optimal solution because of the prohibitive complexity when $n = 50$. In Fig. 2.3 we observe that the projection-based expedient algorithm performs better than the efficacy-based expedient algorithm for some values of m, while the latter performs better for other values of m. We also observe that the projection-based n-path greedy algorithm does not always perform better than the efficacy-based greedy algorithm. Therefore we cannot make an a-priori comparison between these algorithms. However, we note that the efficacy-based n-path greedy solution

Table 2.1: Average runtime of proposed algorithms

Algorithms		Average Runtime (sec)	
		$n = 20$, $m = 10$	$n = 50$, $m = 25$
Efficacy based	Expedient	0.0003	0.0005
	Greedy	0.008	0.15
	n-path Greedy	0.22	10.93
	Backtraced n-path	0.25	12.76
Projection based	Expedient	0.0003	0.001
	Greedy	0.01	0.13
	n-path Greedy	0.2	6.21

and the backtraced n-path solution perform better than the rest of the algorithms. The upper bound in this case is not tight. Table 2.1 shows a comparison of average execution time of different proposed algorithms on an Intel Core i3 @ 2.10 GHz machine with 4GB RAM. From the table we observe that the projection-based algorithms have similar average runtime as the efficacy-based algorithms.

2.6.2 5 by 5 grid model

Next we apply the proposed algorithms to a 5 by 5 grid model where the variance between two points is given in (2.83). We increased the parameter β from 0.5 to 8.0 in three steps to evaluate the solutions. From Fig. 2.4–2.6 we observe that the projection-based expedient solution is not always monotonically increasing. Since the dimension of the upper bounding subspace $\mathbf{F}^*$ increases as m is increased, the dimension of the subspace closest to $\mathbf{F}^*$ also increases. As a result the solution subspace for $(m+1)$ sensors does not always contain the solution subspace for m sensors. Thus the efficacy obtained by projection-based expedient algorithm is not always monotonic with increasing m.

We also note that the projection-based greedy and the n-path greedy algorithm perform better than the efficacy-based expedient solution, but perform worse than the efficacy-based greedy solution in Fig. 2.4. But the projection-base n-path greedy performs better than efficacy-based greedy solution in Fig. 2.5, and performs as good as the efficacy-based n-path

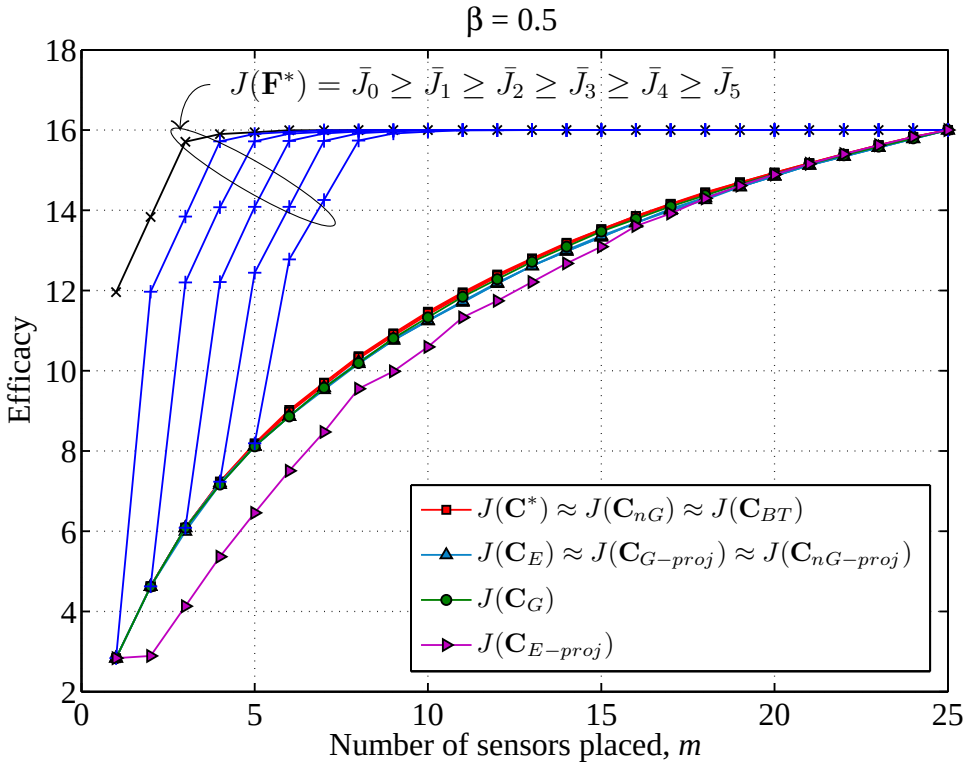


Figure 2.4: $\beta = 0.5$: Efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$ compared to the optimal efficacy $J(\mathbf{C}^*)$

greedy and backtraced solutions in Fig. 2.6. The efficacy-based n-path greedy solution and the backtraced n-path solution are almost indistinguishable from the optimal solution in Fig. 2.4–2.5. In Fig. 2.6, the backtraced algorithm performs slightly worse than the efficacy-based n-path greedy algorithm. The family of upper bounds from Section 2.4 is not tight in Fig. 2.4–2.6. However, the bounds get tighter as β increases.

2.6.3 IEEE 57-bus test system

We next apply the proposed sensor placement algorithms to the IEEE 57-bus test system [65]. Traditional state estimators (SE) in the power grid utilize the redundant measurements taken by supervisory control and data acquisition (SCADA) systems, and the states are estimated using iterative algorithms [4]. With the advent of phasor technology, time synchronized

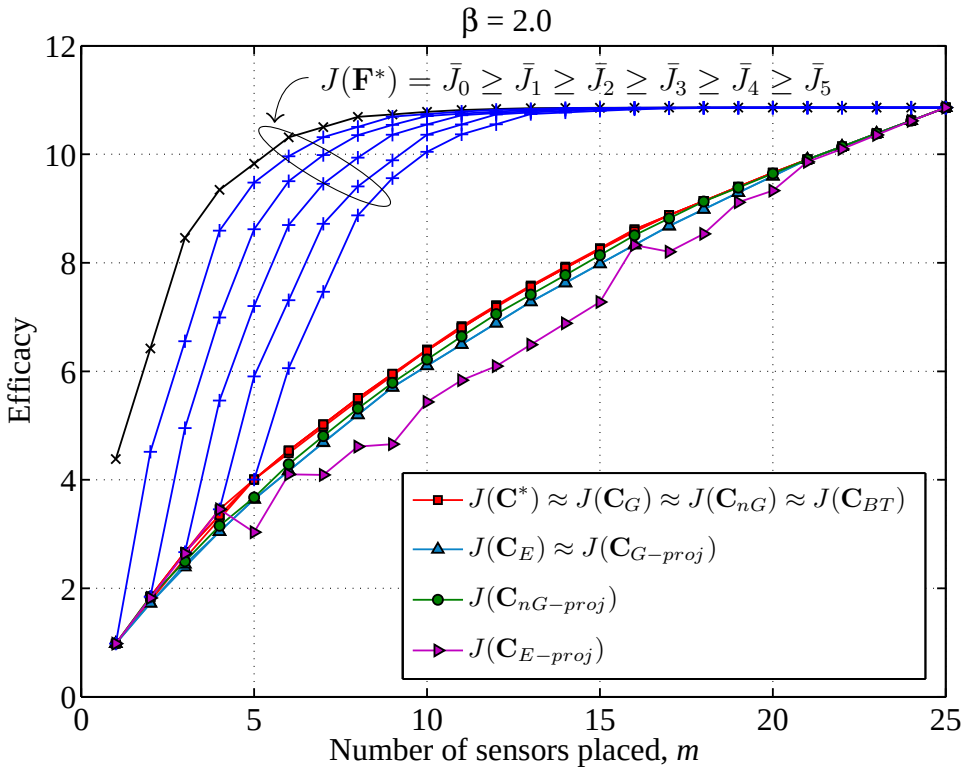


Figure 2.5: $\beta = 2.0$: Efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$ compared to the optimal efficacy $J(\mathbf{C}^*)$

measurements can be obtained using phasor measurement units (PMUs). The PMUs can directly measure the states at the PMU-installed buses, and the states of all the connected buses (given enough channels are available) [3]. Thus, the measurement model (2.4) can be used as the PMU measurement model for the state estimation problem in the power grid. The storage and computational costs of the SE approaches may be further reduced by assuming the voltage magnitudes and phase angles are independent [2, 9]. Thus we consider two independent measurement models for PMU placement, one for voltage magnitudes and the other for phase angles. For simulation purposes, without loss of generality, we assume that the conventional measurements of the test bus system are available to us. We further assume that a sensor is always placed at the swing bus [3] and therefore, the swing bus is not considered in our sensor placement algorithms.

For each measurement model, we construct the sample covariance matrix $\mathbf{\Sigma}_{\underline{X}}$ by running

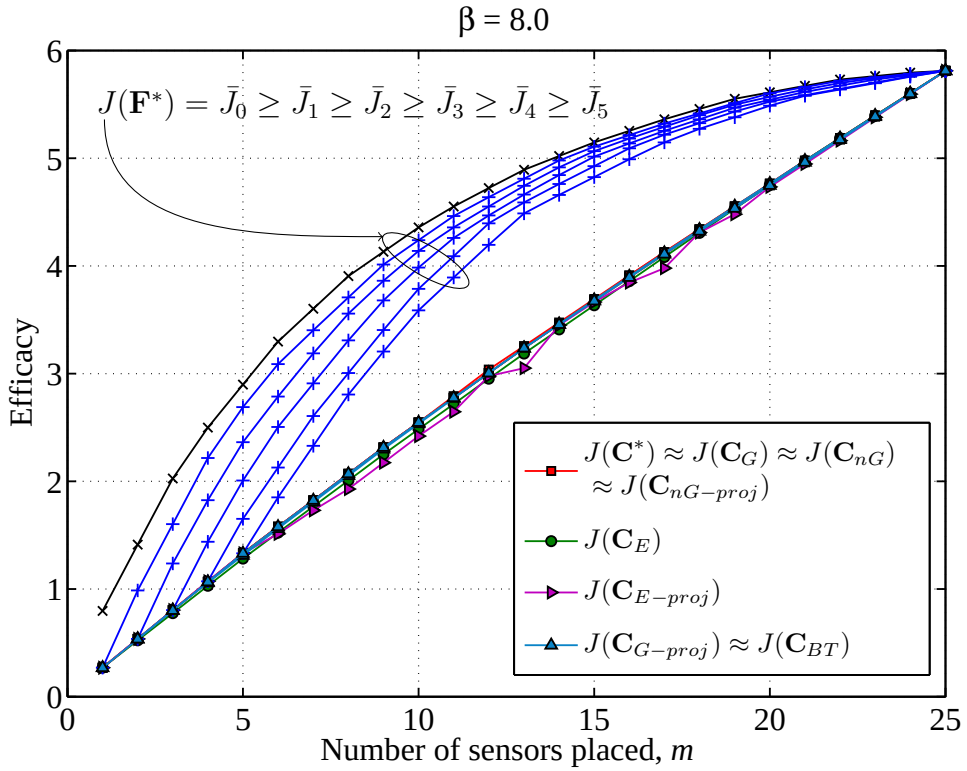


Figure 2.6: $\beta = 8.0$: Efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$ compared to the optimal efficacy $J(\mathbf{C}^*)$

traditional SE algorithms 1000 times. We used the iterative state estimator based on nonlinear AC power flow equations in the MATPOWER software package [66] for this purpose. Since the measurement noise in the traditional state estimator is Gaussian, the estimated state vectors are also Gaussian [6]. In the state estimation process, the standard deviations for voltage magnitudes, bus power injections, and line power flows measurements are 0.01, 0.015, and 0.02, respectively, in accordance with the setups in [66]. The simulations results of our proposed algorithms and nested bounds for voltage magnitude measurement model and phase angle measurement model are shown in Fig. 2.7 and Fig. 2.8, respectively. Again due to the size of the system, the exhaustive search for optimal solution was not possible. But it is observed that, for both measurement models, all of the proposed solutions perform very close to each other.

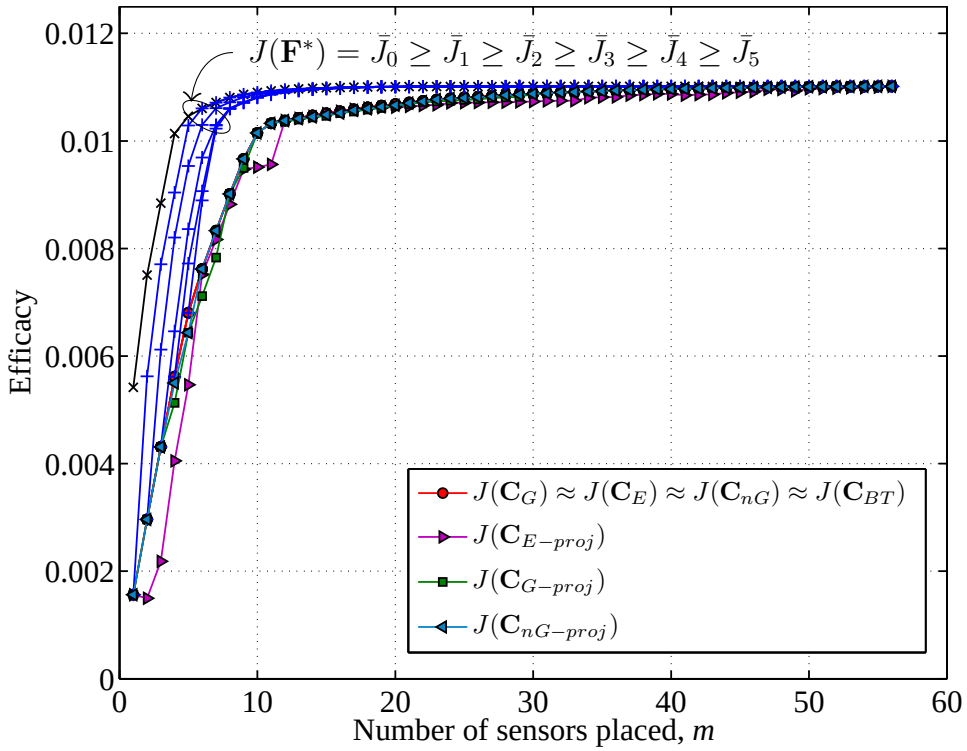


Figure 2.7: IEEE 57-bus test case (voltage magnitudes): efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$

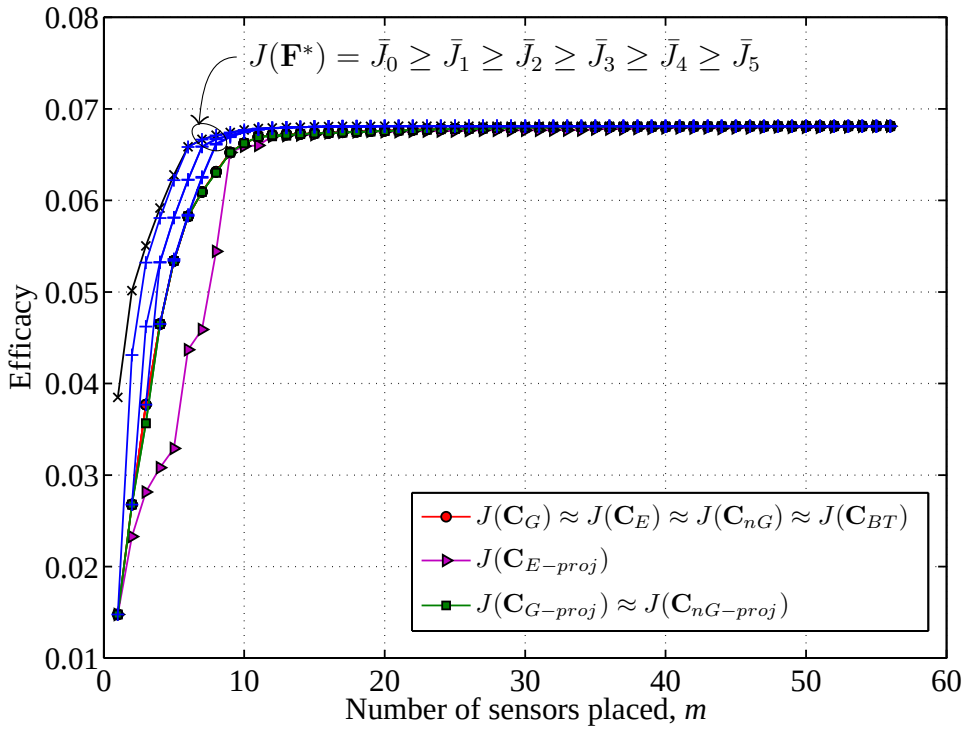


Figure 2.8: IEEE 57-bus test case (phase angles): efficacies of approximate solutions and the upper bound $J(\mathbf{F}^*)$

3

Online Kernel Density Estimation for Outlier Detection

3.1 Largest Normalized Residual Test

The AC state estimation in the power grid is usually formulated as a weighted least squares (WLS) problem. The basic aspects of the WLS estimator and bad data detection are discussed in [4, 9]. In this section, we present a summary of the most commonly used bad data detection method - the largest normalized residual test (r^N_{max} test).

In the WLS estimation method, detection and identification of bad data is done by pro-

cessing the measurement residuals *after* the estimation process. The dectection of bad data from measurement residuals is highly dependent on the availability of well-located, highly redundant measurements. The AC state estimator emplyes the nonlinear measurement model given by [4, 9]

$$\underline{Z} = \underline{h}(\underline{X}) + \underline{N},$$

where $\underline{X}$ is the $n_s \times 1$ system state vector; $\underline{h}(\cdot)$ is the nonlinear vector function relating measurements to the state vector; $\underline{Z}$ is the $n_m \times 1$ measurement vector; and $\underline{N}$ is the $n_m \times 1$ iid measurement error vector. The error vector is assumed to have mean $\underline{0}$ and diagonal covariance $\mathbf{\Sigma}_{\underline{N}}$. The state vector $\underline{x}$ usually consists of the steady state bus voltage magnitudes and phase angles [4]. In the AC state estimation model using SCADA measurements, we usually have $n_m > n_s$.

The WLS estimator then minimizes the following objective function:

$$J(\underline{X}) = \left[\underline{Z} - \underline{h}(\underline{X})\right]^T \mathbf{\Sigma}_{\underline{N}}^{-1} \left[\underline{Z} - \underline{h}(\underline{X})\right]. \tag{3.1}$$

The state estimate $\hat{\underline{X}}$ is obtained using the Gauss-Newton iteration method as shown below [4, 9]:

$$\mathbf{G}\left(\underline{X}^{\ell}\right) \Delta\underline{X}^{\ell+1} = \mathbf{H}\left(\underline{X}^{\ell}\right)^T \mathbf{R}^{-1} \left[\underline{Z} - \underline{h}\left(\underline{X}^{\ell}\right)\right], \tag{3.2}$$

$$\underline{X}^{\ell+1} = \underline{X}^{\ell} + \Delta\underline{X}^{\ell+1}, \tag{3.3}$$

where $\mathbf{G}\left(\underline{X}^{\ell}\right) = \left(\mathbf{H}\left(\underline{X}^{\ell}\right)^T \mathbf{\Sigma}_{\underline{N}}^{-1} \mathbf{H}\left(\underline{X}^{\ell}\right)\right)$ is the $n_s \times n_s$ gain matrix; $\mathbf{H}\left(\underline{X}^{\ell}\right) = \left[\frac{\partial \underline{h}(\underline{X})}{\partial \underline{x}}\right]_{\underline{X}=\underline{X}^{\ell}}$ is the $n_m \times n_s$ Jacobian matrix; ℓ is the iteration number. The Gauss-Newton method generally has linear convergence given the gain matrix $\mathbf{G}\left(\underline{X}^{\ell}\right)$ is not ill-conditioned [67].

Let $\mathbf{G}$ and $\mathbf{H}$ denote the gain and Jacobian matrices, respectively, at the time of the convergence. The covariance matrix of the estimate of the measurement estimate, $\hat{\underline{Z}} = \underline{h}(\hat{\underline{X}})$ is given by

$$\mathbf{\Sigma}_{\hat{\underline{Z}}} = \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T, \tag{3.4}$$

where $\mathbf{H} = \left[\frac{\partial \underline{h}(\underline{X})}{\partial \underline{x}}\right]_{\underline{X}=\hat{\underline{X}}}$.

The measurement residual vector is calculated as the difference between the measurement $\underline{Z}$ and the estimated measurement $\hat{\underline{Z}}$:

$$\underline{R} = \underline{Z} - \hat{\underline{Z}}, \tag{3.5}$$

which is a white Gaussian process of zero mean and has a covariance matrix given by the difference between the measurement error covariance and the measurement estimate covariance [4, 9], i.e.,

$$\mathbf{\Sigma}_{\underline{R}} = \mathbf{\Sigma}_{\underline{N}} - \mathbf{\Sigma}_{\hat{\underline{Z}}}. \tag{3.6}$$

In the largest normalized residual test (r^N_{max} test), $\underline{R}$ is normalized and the largest normalized residual is compared with a threshold, λ:

$$\max \frac{|R_i|}{\sqrt{\mathbf{\Sigma}_{\underline{R}}[i,i]}} \leq \lambda.$$

Usually the threshold is chosen as $\lambda = 3$ [4]. If the i-th measurement violates the threshold, then it is suspected as a bad data, removed from the measurement set and the state estimation process is repeated with the reduced set.

The largest normalized residual test (r^N_{max} test) is summarized in Algorithm 3.1 [4]:

Algorithm 3.1: Largest Normalized Residual Test

Initialize: $\underline{R} = \underline{1}\lambda$.

while $\max(\underline{R}) \geq \lambda$ **do**

- Solve WLS estimator and compute $\underline{R} = \underline{Z} - \underline{h}(\hat{\underline{X}})$.
- Compute normalized residuals $R^N_i = \frac{R_i}{\sqrt{\mathbf{\Sigma}_{\underline{R}}[i,i]}}$, for $i = 1, \ldots, m$.
- Find $k = \arg\max_i R^N_i$.
- If $R^N_k \geq \lambda$, the k-th measurement Z_k is the bad data.
- Set $\underline{Z} = \left[Z_1, \ldots, Z_{k-1}, Z_{k+1}, \ldots, Z_m\right]^T$.
- Set $m \leftarrow m - 1$

end

The performance of the r^N_{max} test is highly dependent on the type of bad data and the measurement configuration [4]. A severe limitation of the r^N_{max} test is that it fails to detect bad data in a measurement if it is a critical (non-redundant) measurement, or its removal does not create a critical measuement in the remainder of the measurement set. In this case the state estimator yields a measurement estimate that will be equal to the measured value [4, 9]. Thus the corresponding measurement residual will be zero. Another shortcoming of the residual test is the detection of bad data in multiple interacting measurements. If multiple bad data with strongly correlated residuals exits, the r^N_{max} test may fail to identify them as many residuals exceed the threshold, and among them some residuals correspond to valid measurements. This is known as the bad data smearing effect.

As seen in the above treatment, the r^N_{max} test requires multiple runs of the state estimator to be able to identify multiple bad measurements, if it can identify at all. When data start to arrive more frequently, this algorithms fails to provide real time information about the quality of the incoming measurements.

Other methods of outlier detection have also been extensively studied in the literature, e.g., hypothesis testing [4, 9], quickest detection using cumulative sum algorithm [39], conditional covariance test [41] etc. However, these methods still depend on the local or global state estimates [39, 41]. Our goal is to design an outlier detection algorithm that can detect bad data without relying on the computationally expensive state estimator.

3.2 Kernel Density Estimation

The probability density function (pdf) is one of the core concepts of mathematical statistics. For any continuous random variable $\underline{X} \in \mathbb{R}^d$, the cumulative distribution function (cdf) is defined by

$$F_{\underline{X}}(\underline{x}) = \Pr(\underline{X} \leq \underline{x}). \tag{3.7}$$

If $F_{\underline{X}}(\underline{x})$ is continuous at $\underline{x}$ then the pdf can be defined as

$$f_{\underline{X}}(\underline{x}) = \frac{d}{d\underline{x}} F_{\underline{X}}(\underline{x}). \tag{3.8}$$

However, in a real-world application scenario the true density $f_{\underline{X}}(\underline{x})$ is usually not known and needs to be estimated from observed data samples using a density estimation technique. The Kernel Density Estimator is one of the most popular and mathematically well studied density estimation tool used for this purpose. The kernel density estimator is a *non-parametric* density estimation method where the data samples solely determine the probability density function $f_{\underline{X}}(\underline{x})$, as opposed to *parametric* methods which impose a known parametric family of distribution and estimate the associated parameters [68]. A kernel function $K : \mathbb{R}^d \to \mathbb{R}$ is any function that satisfies

$$\int_{\mathbb{R}^d} K(\underline{x}) d\underline{x} = 1.$$

For a sample set $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n$ the general multivariate kernel density estimator is defined as [68, 69]

$$\hat{f}_{\underline{X}}(\underline{x}) = \frac{1}{n\,|\mathbf{H}|} \sum_{i=1}^{n} K(\mathbf{H}^{-1}(\underline{x} - \underline{x}_i)). \tag{3.9}$$

Here $\underline{x} = [x_1, x_2, \ldots, x_d]^T$ and $\underline{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$ for $i = 1, 2, \ldots, n$; $K(\underline{x})$ is the kernel function; and $\mathbf{H}$ is a $d \times d$ non-singular bandwidth matrix. Most commonly used kernels are symmetric non-negative *second order* kernels, i.e., they have non-zero second moments and

they satisfy the following conditions:

$$\begin{aligned} K(\underline{x}) &\geq 0, \\ K(\underline{x}) &= K(-\underline{x}), \\ \int_{\mathbb{R}^d} K(\underline{x}) d\underline{x} &= 1, \\ \int_{\mathbb{R}^d} \underline{x} K(\underline{x}) d\underline{x} &= 0, \\ \int_{\mathbb{R}^d} \underline{x}\, \underline{x}^T K(\underline{x}) d\underline{x} &= \mathbf{\Lambda}. \end{aligned}$$

The most common metric used to evaluate the quality of a density estimator is the Mean Integrated Squared Error (MISE) defined as [68, 69]

$$MISE(\hat{f}) = \mathbf{E} \int_{\mathbb{R}^d} [\hat{f}_{\underline{X}}(\underline{x}) - f_{\underline{X}}(\underline{x})]^2 d\underline{x}, \tag{3.10}$$

where $f_{\underline{X}}(\underline{x})$ is the true pdf. The selection of the pair $(\mathbf{H}, K)$ may be tackled as one problem, and in fact, $\mathbf{H}$ may be absorbed into K as a scaling factor. If K is chosen to be any non-negative symmetric kernel there is little to choose between a variety of kernels on the basis of MISE [68, 69]. For this reason, researchers have typically decoupled the choices of K and $\mathbf{H}$, picked K fixed as a non-negative symmetric function. In contrast, significant effort has been given toward finding the optimal bandwidth $\mathbf{H}$ [68, 69]. For our application, we will use the multivariate Gaussian kernel function $K(\underline{x}) = \mathcal{N}(\underline{0}, \mathbf{I})$ throughout this chapter.

3.2.1 Online KDE over a Data Stream

A data stream is an open-ended sequence where large volume of data is arriving continuously. For example, banking transactions, web traffic logs continuously collect large amounts of data. In the power grid the sensors located at the buses take measurements at specific intervals and these measurements arrive at the processing center as a data stream. In these applications processing of the accumulated data in batches is not practical as data

accumulates faster than it can be processed. Any algorithm processing high volume open-ended data streams should meet the following criteria [70]:

1. Each sample must be processed only once.
2. Processing time of each element of the data stream must be small and constant.
3. The algorithm must only use a preallocated amount of main memory.
4. A valid model must be available at every scan of the data stream.
5. The algorithm must produce a model that is equivalent to the one that would be produced by a batch processing algorithm.

The kernel density estimation provides a comprehensive statistical model of the data. However, when applied to a streaming model, the computational requirement of KDE collides with the aforementioned data stream processing requirements. As data arrives continuously it becomes infeasible to evaluate the kernel function at every sample of the collected data. Hence, KDE cannot be applied to data streams in its original form. A naive approach to deal with the growing sample size is to maintain a *window* of data samples. In a simple window model we only evaluate the KDE algorithm on the most recent n samples.

3.2.2 Selection of the Bandwidth Matrix

While several methods have been proposed for selecting suitable bandwidth matrices [69, 71–73], most of these methods are computationally expensive making them less suitable for an online algorithm. For online KDE we use a simple plug-in bandwidth matrix given by [68, 69]

$$\mathbf{H} = a_K \hat{\mathbf{\Sigma}}^{1/2} n^{-1/(d+4)}, \tag{3.11}$$

where $a_K = \{4/(2d+1)\}^{1/(d+4)}$ for multivariate Gaussian kernel [68] and $\hat{\mathbf{\Sigma}}$ is the sample covariance matrix of the data stream calculated in an online fashion. However, the sample covariance matrix is not a very robust statistic as it is sensitive to outliers and fails to capture

the dispersion of datasets with variable spread. To overcome this, we propose clustering the data stream based on the spread of the data and computing the sample covariance matrix for each cluster separately, i.e., we use different bandwidth matrix in different regions of $\mathbb{R}^d$. More precisely, if the data can be divided into r clusters, the variable kernel estimate can be written as [74]

$$\hat{f}_{\underline{X}}(\underline{x}) = \frac{1}{n} \sum_{j=1}^{r} \frac{1}{|\mathbf{H}|_j} \sum_{\underline{x}_i \in S_j} K(\mathbf{H}_j^{-1}(\underline{x} - \underline{x}_i)), \tag{3.12}$$

where $\mathbf{H}_j$ is the estimated bandwidth matrix for cluster S_j. Clustering the data and then summing over all the clusters is only a permutation of the terms in (3.9), and therefore, has no effect on the final estimate. The variable kernel estimate promise substantially improved performance over ordinary kernel estimates for densities with varying behavior in different regions of the space. To exploit the advantages, one must design a good data dependent way of determining the partition [68, 69, 74].

Recently, the notion of clustering data for density estimation has been investigated by Heinz and Seeger in [75]. Their motivation for clustering derives from the system memory constraints which they solve by storing only the centroids of the clusters obtained by the k-means clustering algorithm. However, the k-means algorithm fails to capture the different sizes and shapes of the data [76]. Since Heinze and Seeger were only interested in the memory constraint and used a global bandwidth parameter, this approach worked for their problem. Our motivation comes from the need to choose a more suitable bandwidth parameter that can capture the "local" dispersion of the data. Also when the data is divided into clusters, at every time scan the kernel function needs to re-evaluated only for the data points within one cluster rather than all data points within the window. To capture the dispersion of data in clusters of different sizes and shapes, we present the shared nearest neighbor (SNN) approach proposed by [77] in the next subsection.

3.2.3 Shared Nearest Neighbor Clustering

Several algorithms have been proposed in the literature to handle the problem of data clustering based on density. In particular DBSCAN [78], CURE [79], and Chameleon [80] clustering algorithms have been shown to perform very well for low dimensional data. However, all of the above three algorithms struggle to handle high dimensional data as they use the Euclidean distance between points to separate the clusters. While Euclidean distance is useful in low dimensions, in high dimensional datasets the distance between data points become more uniform making this type of clustering difficult.

A promising way to solve the clustering problem in high dimensional data is a shared nearest neighbor (SNN) approach, first proposed by Jarvis and Patrick in [77], and further improved by Ertöz, Steinback, and Kumar in [81]. In this method two data points are classified as "similar" (i.e., belong to the same cluster) if they share a certain number of nearest neighbors. We introduce the method using the following definitions.

Definition 3.1. *Define the similarity between points $\underline{x}_i$ and $\underline{x}_j$ as*

$$s_{snn}(\underline{x}_i, \underline{x}_j) = |\ell_{nn}(\underline{x}_i) \cap \ell_{nn}(\underline{x}_j)|, \tag{3.13}$$

where $\ell_{nn}(\underline{x}_i)$ and $\ell_{nn}(\underline{x}_j)$ are the k-nearest neighbor lists of $\underline{x}_i$ and $\underline{x}_j$, respectively. ■

Definition 3.2. *Shared nearest neighbor density of each data point is defined as the number of neighbors that have an s_{snn} greater a user specified threshold s_{th}, i.e.,*

$$d_{snn}(\underline{x}_i) = \sum_{\underline{x}_j \in \ell_{nn}(\underline{x}_i)} I(s_{snn}(\underline{x}_i, \underline{x}_j) \geq s_{th}), \tag{3.14}$$

where $I(\cdot)$ is the indicator function. ■

Definition 3.3. *The points that have an d_{snn} greater that another user specified parameter d_{th}, are called* core points, *C_P.* ■

Definition 3.4. *For any data point $\underline{x}_i$, the set $B(\underline{x}_i, s_{th})$ is defined as the set containing all*

the data points within radius s_{th} of $\underline{x}_i$, i.e.,

$$B(\underline{x}_i, s_{th}) = \left\{ \underline{x}_j : \underline{x}_j \in \ell_{nn}(\underline{x}_i) \text{ and } s_{snn}(\underline{x}_i, \underline{x}_j) \geq s_{th} \right\}. \tag{3.15}$$

If $\underline{x}_i$ is a core points, then all the points in $B(\underline{x}_i, s_{th})$ are placed in the same cluster with $\underline{x}_i$.

■

The pair point similarity measure has the advantage of automatic scaling depending on the data density, i.e., the neighborhoods expand where the data points are widely spread and the neighborhood shrinks where the data points are dense. Since there is no globally fixed distance threshold, the algorithm can produce clusters of different sizes, shapes and densities [77].

We can easily extend this clustering algorithm to cluster over data streams. When a new data point arrives, there are three possibilities:

1. It belongs to an existing cluster.
2. It represents the beginning of a new cluster.
3. It is an outlier.

In a data stream it is very difficult to distinguish between the last two cases. Thus, we keep these types of data points in the data window rather than discarding them right away and handle the last two cases the same way, by assigning the incoming data point to a new cluster. Similar to the offline SNN algorithm, we can initialize the online SNN algorithm by finding the k nearest neighbors first. However, since this new data point does not belong to any of the old data points' nearest neighbor list, we need to update the neighbor list of the k nearest neighbors of the new incoming data point. This can be easily done by comparing the largest distance of the old data points neighbor list to the distance from the incoming data point. Therefore, when finding the nearest neighbors for the data points, we not only need to save the index of the neighbors but also the corresponding distance measure. Then at most k old data points need to have their neighbor list updated for each incoming data point

in the stream. Once the neighbor lists are updated we can proceed to compute the s_{snn} and then d_{snn} of the incoming data point. Comparing the d_{snn} with the predefined thresholds s_{th} and d_{th} we have three possible cases:

Case 1: If the new data point has d_{snn} greater than the threshold d_{th}, then classify the data point as a *core point*. If the new core point is within radius s_{th} of another core point, then they belong in the same cluster. Otherwise, the new core point indicates the beginning of a new cluster. Thus, a new cluster is formed containing the new core points and its neighbors within radius s_{th}.

Case 2: If the new data point has d_{snn} less than d_{th} but it is within radius s_{th} of a core point then the new data point is put in the same cluster as the core point.

Case 3: Otherwise the new data is labeled as possible outlier and it is put into a cluster by itself.

We can summarize the online shared nearest neighbor clustering in Algorithm 3.2. The computational complexity of the shared nearest neighbor clustering algorithm is determined by the nearest neighbor search step. For example, using the kd-tree algorithm [82] for k nearest neighbor search has complexity $\mathcal{O}(kG(d)\log n)$, where $G(d)$ is some function exponential in d. Note that worst case complexity is always bounded by $\mathcal{O}(kdn)$.

As mentioned earlier, for an online clustering algorithm it is very difficult to distinguish between the beginning of a new cluster and an outlier. The only way this algorithm can identify the beginning of a new cluster is after observing at least d_{th} number of data points from the new cluster. Also after d_{th} number of data points have been observed from a new cluster, it is very likely that the old data points belonging to the clusters could now be classified as core points as their neighbor list is updated with the arrival of the new data points. This could be easily performed by storing the SNN similarity matrix in the memory and updating the similarity measure of the neighbors of the new data point. This step can be performed by a background process and the online algorithm can fetch the results at appropriate intervals.

Algorithm 3.2: Online Shared Nearest Neighbor Clustering Algorithm

Initialize: $C_P = \{\}$
for $n = k+1, k+2, \ldots$ **do**
  Compute k nearest neighbor list $\ell_{nn}(\underline{x}_n)$.
  Update neighbor lists of $\underline{x}_j \in \ell_{nn}(\underline{x}_n)$, if necessary.
  for *each* $\underline{x}_j \in \ell_{nn}(\underline{x}_n)$ **do**
    compute $s_{snn}(\underline{x}_n, \underline{x}_j) = |\ell_{nn}(\underline{x}_n) \cap \ell_{nn}(\underline{x}_j)|$.
  end
  compute $d_{snn}(\underline{x}_n) = \sum_{\underline{x}_j \in \ell_{nn}(\underline{x}_n)} I(s_{snn}(\underline{x}_n, \underline{x}_j) \geq s_{th})$.
  if $d_{snn}(\underline{x}_n) \geq d_{th}$ **then**
    if $B(\underline{x}_n, s_{th}) \cap C_P \neq \emptyset$ **then**
      $\underline{x}_n$ and $B(\underline{x}_n, s_{th}) \cap C_P$ belong to the same cluster.
    else
      $\underline{x}_n$ and $B(\underline{x}_n, s_{th})$ form a new cluster.
    end
    $C_P = C_P \cup \{\underline{x}_n\}$.
  else
    if $B(\underline{x}_n, s_{th}) \cap C_P \neq \emptyset$ **then**
      $\underline{x}_n$ and $B(\underline{x}_n, s_{th}) \cap C_P$ belong to the same cluster.
    else
      classify $\underline{x}_n$ as a possible outlier and put in a new cluster.
    end
  end
end

3.2.4 Outlier Detection over Data Streams

An outlier is a data point that appears different from the other data points within the sample space. In sensor networks an outlier may be generated by faulty sensors or an event in the network. An outlier may also be caused by injections by network attackers. In the power grid state estimation process the sensor measurements from different parts of the network are collected and processed to generate an estimate of the system states. Since the state estimator is an indispensable tool in the grid monitoring process it is imperative that any outlier present in the incoming sensor measurements be detected and identified before the data is fed into the state estimator.

Since the outliers are observations that have very low probability of occurrence, the online

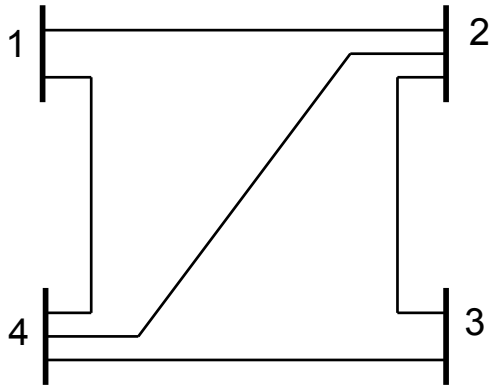


Figure 3.1: Sample 4 bus system

density estimator can be used to identify potential outliers based on their probability. We can classify the outliers in the power system sensor measurements into two categories – spatial outliers and temporal outliers.

Spatial outlier A spatial outlier is a measurement observed at a bus that differs significantly given the measurements made in the spatial neighborhood. Spatial neighborhoods are defined based on the adjacency of the buses. We can model the spatial dependency of the measurements by estimating a joint density function over a bus and its neighbors. We can the compute the conditional probability of a new observation at the bus given the measurements at its neighbors from the joint density function. If the conditional probability is below a specified threshold then the new incoming data is classified as a spatial outlier.

Example 3.1. *Let us consider the simple 4 bus system shown in fig. 3.1. Here bus 1 is connected to bus 2 and 4, thus bus 2 and 4 are in the spatial neighborhood of bus 1. If the random variables* X_i, $i = 1, \ldots, 4$ *denote the measurements at the buses, then we can estimate the joint density function* $\hat{f}_{\underline{X}}(\underline{x})$, $\underline{X} = [X_1, X_2, X_4]^T$ *using the online KDE described in Section 3.2. Then we can compute the conditional probability of a new measurements* x_1

at bus 1 given the measurements x_2 and x_4 at bus 2 and 4, respectively, as

$$\Pr\left(X_1 \in [x_1 - r, x_1 + r] | X_2 \in [x_2 - r, x_2 + r], X_4 \in [x_4 - r, x_4 + r]\right) = \frac{\int_{x_1-r}^{x_1+r} \int_{x_2-r}^{x_2+r} \int_{x_4-r}^{x_4+r} \hat{f}_{\underline{X}}(\underline{x}) d\underline{x}}{\int_{-\infty}^{\infty} \int_{x_2-r}^{x_2+r} \int_{x_4-r}^{x_4+r} \hat{f}_{\underline{X}}(\underline{x}) d\underline{x}}. \tag{3.16}$$

The new measurement x_1 is a spatial outlier if the conditional probability computed in (3.16) *is less than a threshold γ.* ■

The spatial outliers are a local anomaly as they are only dependent on the neighbor measurements. This method of outlier detection requires one hop communication between the buses.

Temporal outlier A temporal outlier is an observation at a bus where the deviation of the current observation from the previous observation differs significantly from the deviations seen so far in the stored history of the bus. That is, we assume that there is temporal dependency between consecutive measurements. This dependency can be modeled as a Markov process [63] where the current state of the bus is solely dependent on the previous state in a first order Markov process.

Again, we can compute the transition probability in an online fashion using online KDE. We can estimate the joint density $\hat{f}_{\underline{X}}(\underline{x})$, $\underline{X} = [X[t], X[t-1]]^T$, where t is the time index, and then compute the transition probability using a similar formula as (3.16). A measurement is classified as a temporal outlier if it has a transition probability less than a user defined threshold.

The spatial and temporal dependency of the measurements can be integrated to provide a spatio-temporal outlier. One way to accomplish this is to first identify a spatial (or temporal) outlier and then check whether this outlier was also experienced is the temporal (or spatial) model [83].

Algorithm 3.3 summarizes the online outlier detection on a bus. The running time of this

algorithm is dominated by the k nearest neighbor search in the SNN clustering step when the number of points where density is estimated is small.

Algorithm 3.3: OUTLIER DETECTION USING KDE

for *each incoming data point* **do**
  Compute the conditional probability ($\Pr[x|x_{neighbor}]$ for spatial and $\Pr[x[t]|x[t-1]]$ for temporal outliers).
  if *conditional probability* $<$ *threshold,* γ **then**
    Declare Outlier
  end
  Classify the new data into a cluster S_j using SNN clustering.
  Update the estimated bandwidth $\mathbf{H}_j$ for the cluster.
  Re-evaluate the kernel function for the data points in S_j using the updated $\mathbf{H}_j$.
  Update the joint density estimate $\hat{f}(\underline{X})$ using (3.12).
end

3.2.5 Simulation Results

To evaluate the performance of the proposed KDE based outlier detection algorithm we implemented the algorithm in the IEEE 14-bus test system [65]. The network diagram of the test system is shown in Fig. 3.2. We compare the performance of our algorithm with one of the most commonly used bad data identification method, the largest normalized residual test (the r^N_{max} test). The limitation of the r^N_{max} test is that it fails to identify bad data in critical measurements [4]. Also, sometimes it may fail to identify multiple interacting bad data. As shown below our method is able to overcome these limitations.

We used the state estimator in the MATPOWER softwater [66] as a tool for the largest normalized residual test (r^N_{max} test). The measurement configuration used for the simulations is given in Table 3.1.

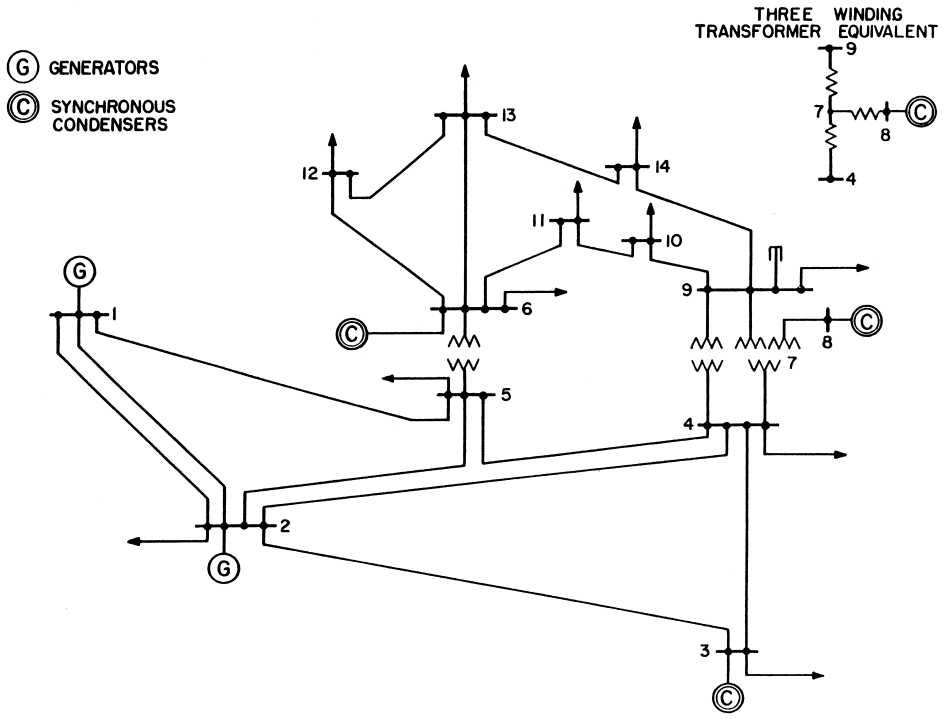


Figure 3.2: IEEE 14 bus system [65]

Table 3.1: Measurement configuration for IEEE 14-bus test system

Power injection	At buses 1, 2, 3, 8
Power flow	At branches 1-2, 1-5, 2-4, 2-5, 4-5, 4-7, 4-9, 5-6, 6-11, 6-13, 9-10, 9-14, 12-13
Voltage magnitude	At buses 3, 6, 8, 10, 14

3.2.5.1 Bad data in a critical measurement

Under the measurement configuration described in Table 3.1, P_{4-7} is a critical measurement, i.e., the removal of this measurement makes the system unobservable. The true value for this flow is $P_{4-7} = 0.2285 p.u.$ The residual for this measurement will always be zero and therefore, any bad data in this measurement is not identifiable by the r^N_{max} test. To apply our online outlier detection technique we create a stream of 1000 samples by adding Gaussian noise $\mathcal{N}(0, 0.0004)$ to the true value. We then randomly inject five bad data points in this data stream. For spatial outlier detector we considered the vector $\underline{X} = [P_{4-7}, P_{4-2}, P_{4-5}, P_{4-9}]^T$. Then we computed the conditional probability $\Pr\{P_{4-7}|P_{4-2}, P_{4-5}, P_{4-9}\}$ and compared it

with the threshold $\gamma = 1\%$. The same threshold was used for temporal outlier detector. The values of the bad measurements at P_{4-7} and the performance of the different bad data detection algorithm is shown in Table 3.2. From the table we see that the proposed online outlier technique was able to identify every bad measurement *before* the data was fed to the state estimator.

Table 3.2: Detection of bad data in a critical measurement P_{4-7} using KDE

Bad data in P_{4-7}	r_{max}^N test	Spatial outlier	Temporal outlier
$0.3152 p.u.$	Not detected	Detected	Detected
$0.0660 p.u.$	Not detected	Detected	Detected
$0.3828 p.u.$	Not detected	Detected	Detected
$0.1477 p.u.$	Not detected	Detected	Detected
$0.3315 p.u.$	Not detected	Detected	Detected

3.2.5.2 Bad data in multiple interacting measurements

Here we consider the occurrence of bad data in two interacting measurements P_1 and P_{1-2}. The actual values of these measurements are $P_1 = 1.9433 p.u.$ and $P_{1-2} = 1.2967 p.u.$ For these measurements we construct a vector $\underline{X} = [P_1, P_{1-2}, P_2, P_{1-5}, P_{2-4}, P_{5-4}, P_{5-6}]^T$ for spatial outlier detection and randomly inject bad data in a data stream of 1000 samples. The results of the bad data detection methods are shown in Table 3.3. The results show that the r_{max}^N fails sometimes when there are multiple interacting bad data and even when it is able to detect the outliers it takes two passes of the state estimator before both bad measurements are identified. On the other hand our proposed method successfully detects all the injected bad data without the results from the state estimator.

Table 3.3: Bad data in a interacting measurement P_1 and P_{1-2} using KDE

Bad data	r^N_{max} test	Spatial outlier	Temporal outlier
$P_1 = -1.0431 p.u.$, $P_{1-2} = -0.0246 p.u.$	Detected	Detected	Detected
$P_1 = -2.2014 p.u.$, $P_{1-2} = -2.2022 p.u.$	Not detected	Detected	Detected
$P_1 = -1.9009 p.u.$, $P_{1-2} = -1.7306 p.u.$	Not detected	Detected	Detected
$P_1 = 2.9055 p.u.$, $P_{1-2} = -1.6076 p.u.$	Detected	Detected	Detected
$P_1 = 0.0896 p.u.$, $P_{1-2} = -0.5531 p.u.$	Not detected	Detected	Detected

4

Online Least-squares One-class Support Vector Machine

4.1 Background

In Chapter 3 we saw that the KDE based online outlier detection method provides improved performance over the traditional r^N_{max} test used in the power grid. However, estimation of probability density over the entire support of the data becomes computationally expensive when the dimension of the input data (number of neighboring measurements) become larger. To address the computational issues we investigate another approach for outlier detection.

The general idea is to find the regions of the support where the probability density is in some sense large, rather than estimating the density over the entire support. To capture the regions where most of the data are located, Schölkopf et al. proposed the One-class support vector machine (SVM) algorithm in [47]. In this section we will provide a brief introduction of the standard One-class SVM and then discuss its least squares variation.

4.1.1 Standard One-class Support Vector Machine

The general idea behind the standard one-class support vector machine (OC-SVM) is to find the regions of the support where the probability density is in some sense large. Several formulations for the standard one-class SVM has been proposed in the literature [47, 48]. One approach tightly fits the training objects in the feature space inside a hypersphere of minimum volume [48], while another considers separating the training data from the origin using a hyperplane [47]. These different formulations have been also shown to be equivalent [48, 84]. In this section, we will provide a brief introduction to the standard one-class SVM.

Schlkopf et al. [47] formulate the one-class SVM as an optimization problem that searches for a hyperplane parameterized by a weight vector $\underline{w}$ and a bias ρ, such that most of the training objects reside on one side of the hyperplane, and the distance $\rho/\|\underline{w}\|$ between the origin and the hyperplane is maximized. For training data $\underline{x}_1, \ldots, \underline{x}_n \in \mathbb{R}^d$, the optimization problem can be stated as

$$\begin{aligned} \min_{\underline{w},\rho,\xi_j} \; & \frac{1}{2}\|\underline{w}\|_2^2 - \rho + C\sum_{j=1}^{n}\xi_j \\ \text{subject to} \quad & \underline{w}^T\underline{\phi}(\underline{x}_j) \geq \rho - \xi_j, \quad j = 1, \ldots, n \\ & \xi_j \geq 0, \quad j = 1, \ldots, n \end{aligned} \tag{4.1}$$

where $\underline{\phi}(\cdot) : \mathbb{R}^d \to \mathcal{H}$ is a mapping to a high dimensional feature space such that the dot product in the image of $\underline{\phi}$ can be computed by evaluating a kernel function, i.e., $k(\underline{x}, \underline{y}) =$

$\underline{\phi}(\underline{x})^T\underline{\phi}(\underline{y})$.The parameter C is predefined and controls the fraction of possible outliers [47] and ξ_j are non-zero slack variables. The obtained hyperplane is defined as $f(\underline{x}) = \underline{w}^T\underline{\phi}(\underline{x}) - \rho = 0$.

The dual of the optimization problem (4.1) is given by

$$\min_{\underline{\alpha}} \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k(\underline{x}_i, \underline{x}_j) \tag{4.2}$$

$$\text{subject to} \quad 0 \leq \alpha_j \leq C, \quad j = 1, \ldots, n \tag{4.3}$$

$$\sum_{j=1}^{n}\alpha_j = 1, \tag{4.4}$$

where α_j are the Lagrange multipliers. The values of α_j can be obtained using a standard quadratic program solver. The bias term ρ can be obtained from $f(\underline{x}_s) = 0$, where $\underline{x}_s$ denotes one of the support vectors such that $0 < \alpha_s < C$. The weight vector in the primal problem (4.1) is given by

$$\underline{w} = \sum_{j=1}^{n}\alpha_j\underline{\phi}(\underline{x}_j) = \mathbf{\Phi}\underline{\alpha}, \tag{4.5}$$

where $\mathbf{\Phi} = [\underline{\phi}(\underline{x}_1), \ldots, \underline{\phi}(\underline{x}_n)]$, and the separating hyperplane becomes

$$f(\underline{x}) = \sum_{j=1}^{n}\alpha_j k(\underline{x}_j, \underline{x}) - \rho = 0. \tag{4.6}$$

In [48], Tax and Duin give an alternative formulation for the one-class SVM, which is also called *support vector data description*. The principle idea in this approach is to find a minimum volume hypersphere in the feature space that encloses all the training objects. The center $\underline{a}$ and radius R of the hypersphere is obtained by solving the constrained optimization

problem:

$$\min_{R,\underline{a}} R^2 + C\sum_j \xi_j \tag{4.7}$$

$$\text{subject to } \left\|\underline{\phi}(\underline{x}_j) - \underline{a}\right\|^2 \leq R^2 + \xi_j. \tag{4.8}$$

The dual problem to the optimization problem (4.7) is given by:

$$\max_{\underline{\alpha}} \sum_j \alpha_j k(\underline{x}_j, \underline{x}_j) - \sum_{i,j} \alpha_i \alpha_j k(\underline{x}_i, \underline{x}_j) \tag{4.9}$$

$$\text{subject to } \sum_j \alpha_j = 1, \tag{4.10}$$

$$0 \leq \alpha_j \leq C, \quad j = 1, \ldots, n \tag{4.11}$$

$$\xi_j \geq 0, \tag{4.12}$$

where α_j denote the Lagrange multipliers. Note that (4.2) and (4.9) are equivalent when $k(\underline{x}_i, \underline{x}_j) = 1$. Optimization problem (4.9) can be solved using quadratic programming and the obtained center of the sphere can be written as:

$$\underline{a} = \sum_j \alpha_j \underline{\phi}(x_j). \tag{4.13}$$

The radius R can be computed from $\left\|\underline{\phi}(\underline{x}_s) - a\right\|^2 = R^2$, where $\underline{x}_s$ denotes one of the support vectors. The decision function then becomes

$$g(\underline{x}) = \text{sgn}(R^2 - \|\phi(\underline{x}) - \underline{a}\|^2). \tag{4.14}$$

4.1.2 Least-Squares One-class SVM

The hyperplane obtained in the standard one-class SVM provides a boundary of the region that encloses a given fraction of the training objects, however, it does not well reflect the

distribution of the training objects within the region. The least-squares (LS) version of the standard one-class SVM was proposed by Choi in [49] using a quadratic loss function and an equality constraint. The LS one-class SVM gives a hyperplane that maximizes the distance from the origin and minimizes the distance to the training objects in least squares sense. The distance from this hyperplane may then be used as a measure to determine which objects resemble the training objects better than others. The optimization problem for LS one-class SVM is formulated as

$$\min\ J = \frac{1}{2}\left\|\underline{w}\right\|^2 - \rho + \frac{C}{2}\left\|\underline{\xi}\right\|^2, \tag{4.15}$$

$$\text{subject to } \underline{w} = \mathbf{\Phi}\underline{\alpha}, \tag{4.16}$$

$$\underline{\xi} = \underline{1}_n\rho - \mathbf{\Phi}^T\underline{w}, \tag{4.17}$$

where $\underline{1}_n$ is a $n \times 1$ vector of all ones. Substituting (4.16)–(4.17) into (4.15), we have

$$J = \frac{1}{2}\underline{\alpha}^T\mathbf{K}\underline{\alpha} - \rho + \frac{C}{2}\left\|\underline{1}_n\rho - \mathbf{K}\underline{\alpha}\right\|^2, \tag{4.18}$$

where $\mathbf{K}$ denotes the kernel matrix with entries $\mathbf{K}[i,j] = k(\underline{x}_i, \underline{x}_j) = \underline{\phi}(\underline{x}_i)^T\underline{\phi}(\underline{x}_j)$. Taking derivative of (4.18) with respect to $\underline{\alpha}$ and ρ, and setting equal to zero yields:

$$\frac{\partial J}{\partial \underline{\alpha}} = \mathbf{K}\underline{\alpha} - C\mathbf{K}(\underline{1}_n\rho - \mathbf{K}\underline{\alpha}) = \underline{0}_n \qquad \Longrightarrow \left(\mathbf{K} + \frac{\mathbf{I}_n}{C}\right)\underline{\alpha} - \underline{1}_n\rho = \underline{0}_n \tag{4.19}$$

$$\frac{\partial J}{\partial \rho} = -1 + C\underline{1}_n^T(\underline{1}_n\rho - \mathbf{K}\underline{\alpha}) = 0 \qquad \Longrightarrow -\mathbf{K}\underline{\alpha} + \underline{1}_n^T\underline{1}_n\rho = \frac{1}{C} \tag{4.20}$$

Combining (4.19) and (4.20), we have,

$$\begin{bmatrix} \underline{1}_n^T\underline{1}_n & -\underline{1}_n^T\mathbf{K} \\ -\underline{1}_n & \mathbf{K} + \mathbf{I}_n/C \end{bmatrix} \begin{bmatrix} \rho \\ \underline{\alpha} \end{bmatrix} = \begin{bmatrix} 1/C \\ \underline{0} \end{bmatrix}, \tag{4.21}$$

where $\underline{0}_n$ is a $n \times 1$ vector containing all zeros and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Applying block matrix inversion lemma [85], we obtain

$$\rho = \left(\underline{1}_n^T \left(\mathbf{K} + \frac{\mathbf{I}_n}{C} \right)^{-1} \underline{1}_n \right)^{-1}, \tag{4.22}$$

$$\underline{\alpha} = \left(\underline{1}_n^T \left(\mathbf{K} + \frac{\mathbf{I}_n}{C} \right)^{-1} \underline{1}_n \right)^{-1} \left(\left(\mathbf{K} + \frac{\mathbf{I}_n}{C} \right)^{-1} \underline{1}_n \right). \tag{4.23}$$

The obtained hyperplane is then given by $f(\underline{x}) = \underline{w}^T \phi(\underline{x}) - \rho = \underline{\alpha}^T \underline{k} - \rho = 0$, where $\underline{k}$ denotes a vector with entries $k(\underline{x}_j, \underline{x})$, $j = 1, \ldots, n$.

4.2 Sparse Online LS One-class SVM

In an online application a sparse solution for the LS one-class SVM is desirable so that instead of storing the entire history of training set, the solution can be stored in a compact form. Sparsity also reduces the time to train data as the number of linear equations is reduced. It can also achieve good solutions given that the support vectors are chosen judiciously. However the LS one-class SVM in Section 4.1.2 does not introduce sparsity by itself due to the quadratic loss function in the objective function (4.15). Several approaches to sparsification of kernel-based solutions have been proposed in the literature [50–53]. In this paper, we utilize the approximate linear dependence (ALD) criterion proposed by Engel et al. in[51] to induce sparsity in the LS one-class SVM solution. The ALD cost criterion was first introduced in [51] to induce sparsity in Kernel Recursive Least Squares (KRLS) algorithm in a supervised learning environment. In this section, we extend the application of ALD cost to an unsupervised learning environment in least-squares one-class SVM.

In an online outlier detection scheme we sequentially process a stream of incoming data points. As more and more data become available the memory and processing requirement increases indefinitely. We adopt a dictionary for the sparsity requirement [51]. Let the dictionary at time step $n-1$ be $\mathcal{X}_{\mathcal{D}}^{(n-1)} = \{\underline{x}_{D,1}, \ldots, \underline{x}_{D,m}\}$ where $m < n$; $\underline{x}_{D,j}$ are the support

vectors for the LS one-class SVM. To determine whether to add $\underline{x}_n$ to the dictionary, we compute the following ALD cost [51]:

$$\delta_n = \min_{\underline{\beta}_n} \left\| \mathbf{\Phi}_{D,(n-1)} \underline{\beta}_n - \underline{\phi}(\underline{x}_n) \right\|^2, \tag{4.24}$$

where $\mathbf{\Phi}_{D,(n-1)} = [\underline{\phi}(\underline{x}_{D,1}), \ldots, \underline{\phi}(\underline{x}_{D,m})]$. Here δ_n is the distance of $\underline{x}_n$ to the linear span of the dictionary in the feature space. By expanding the inner product and substituting $\underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}') = k(\underline{x}, \underline{x}')$, we can rewrite (4.24) as

$$\delta_n = \min_{\underline{\beta}_n} \{ \underline{\beta}_n^T \mathbf{K}_{D,(n-1)} \underline{\beta}_n - 2 \underline{\beta}_n^T \underline{k}_n + k_{nn} \}, \tag{4.25}$$

where $\mathbf{K}_{D,(n-1)}[i,j] = k(\underline{x}_{D,i}, \underline{x}_{D,j})$; $\underline{k}_n[i] = k(\underline{x}_{D,i}, \underline{x}_n)$ and $k_{nn} = k(\underline{x}_n, \underline{x}_n)$. Solving (4.25) yields

$$\underline{\beta}_n = \mathbf{K}_{D,(n-1)}^{-1} \underline{k}_n, \tag{4.26}$$

$$\delta_n = k_{nn} - \underline{k}_n^T \mathbf{K}_{D,(n-1)}^{-1} \underline{k}_n. \tag{4.27}$$

If δ_n is less than an accuracy parameter γ, then $\underline{\phi}(\underline{x}_n)$ can be approximated by some linear combination of the present dictionary members within a squared error γ. Defining $\mathbf{\Phi}_n = [\underline{\phi}(\underline{x}_1), \ldots, \underline{\phi}(\underline{x}_n)]$ we can then write

$$\mathbf{\Phi}_n \approx \mathbf{\Phi}_{D,(n)} \mathbf{B}_n^T, \tag{4.28}$$

where $\mathbf{B}_n[i,j] = \underline{\beta}_i[j]$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Then from (4.5), we can write

$$\underline{w}_n = \mathbf{\Phi}_n \underline{\alpha}_n \approx \mathbf{\Phi}_{D,(n)} \mathbf{B}_n^T \underline{\alpha}_n = \mathbf{\Phi}_{D,(n)} \hat{\underline{\alpha}}_n = \hat{\underline{w}}_n, \tag{4.29}$$

where $\hat{\underline{\alpha}}_n = \mathbf{B}_n^T \underline{\alpha}_n$ is the $m \times 1$ approximation of the $n \times 1$ vector $\underline{\alpha}_n$. We can now rewrite

optimization problem (4.15) as an online optimization problem as

$$\min\ J_n = \min_{\underline{\hat{w}}_n, \rho_n, \underline{\xi}_n} \frac{1}{2} \left\|\underline{\hat{w}}_n\right\|^2 - \rho_n + \frac{C}{2} \left\|\underline{\xi}_n\right\|^2 \tag{4.30}$$

$$\text{subject to } \underline{\hat{w}}_n = \mathbf{\Phi}_{D,(n)} \underline{\hat{\alpha}}_n, \tag{4.31}$$

$$\underline{\xi}_n = \underline{1}_n \rho_n - \mathbf{\Phi}_n^T \underline{\hat{w}}_n. \tag{4.32}$$

Substituting $\underline{\hat{w}}_n$ and $\underline{\xi}_n$ into (4.30), we have

$$J_n = \frac{1}{2} \underline{\hat{\alpha}}_n^T \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n - \rho_n + \frac{C}{2} \left\|\underline{1}_n \rho_n - \mathbf{B}_n \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n\right\|^2. \tag{4.33}$$

Taking the derivatives of (4.33) with respect to $\underline{\hat{\alpha}}_n$ and ρ_n, and setting them equal to zero, we have

$$\begin{aligned} \frac{\partial J_n}{\partial \underline{\hat{\alpha}}_n} &= \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n - C \mathbf{K}_{D,(n)} \mathbf{B}_n^T (\underline{1}_n \rho_n - \mathbf{B}_n \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n) = \underline{0}_m \\ &\Longrightarrow (\mathbf{I}_m / C + \mathbf{B}_n^T \mathbf{B}_n \mathbf{K}_{D,(n)}) \underline{\hat{\alpha}}_n - \mathbf{B}_n^T \underline{1}_n \rho_n = \underline{0}_m, \end{aligned} \tag{4.34}$$

and

$$\begin{aligned} \frac{\partial J_n}{\partial \rho_n} &= -1 + C \underline{1}_n^T (\underline{1}_n \rho_n - \mathbf{B}_n \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n) = 0 \\ &\Longrightarrow \underline{1}_n^T \underline{1}_n \rho_n - \underline{1}_n^T \mathbf{B}_n \mathbf{K}_{D,(n)} \underline{\hat{\alpha}}_n = 1/C. \end{aligned} \tag{4.35}$$

Combining (4.34) and (4.35), we obtain the following set of linear equations:

$$\begin{bmatrix} \underline{1}_n^T \underline{1}_n & -\underline{1}_n^T \mathbf{B}_n \mathbf{K}_{D,(n)} \\ -\mathbf{B}_n^T \underline{1}_n & \mathbf{I}_m / C + \mathbf{B}_n^T \mathbf{B}_n \mathbf{K}_{D,(n)} \end{bmatrix} \begin{bmatrix} \rho_n \\ \underline{\hat{\alpha}}_n \end{bmatrix} = \begin{bmatrix} 1/C \\ \underline{0} \end{bmatrix}. \tag{4.36}$$

Defining

$$\underline{u}_n \triangleq \mathbf{B}_n^T \underline{1}_n, \tag{4.37}$$

$$\mathbf{T}_n \triangleq \mathbf{B}_n^T \mathbf{B}_n, \tag{4.38}$$

$$\mathbf{P}_n \triangleq \mathbf{I}_m / C + \mathbf{T}_n \mathbf{K}_{D,(n)}, \tag{4.39}$$

and applying block matrix inversion lemma [85], we can compute ρ_n and $\underline{\hat{\alpha}}_n$ as:

$$\rho_n = \left(C \underline{1}_n^T \left(\mathbf{I}_n - \mathbf{B}_n \mathbf{K}_{D,(n)} \mathbf{P}_n^{-1} \mathbf{B}_n^T \right) \underline{1}_n \right)^{-1}, \tag{4.40}$$

$$\underline{\hat{\alpha}}_n = \mathbf{P}_n^{-1} \underline{u}_n \left(C \underline{1}_n^T \left(\mathbf{I}_n - \mathbf{B}_n \mathbf{K}_{D,(n)} \mathbf{P}_n^{-1} \mathbf{B}_n^T \right) \underline{1}_n \right)^{-1}. \tag{4.41}$$

Note that when there is no sparsity, i.e., $\mathbf{B}_n = \mathbf{I}_n$, (4.40) and (4.41) reduces to (4.22) and (4.23), respectively.

In an online data stream application, ρ_n and $\underline{\hat{\alpha}}_n$ need to be updated at every time step when a new data point becomes available. Using (4.40) and (4.41) as update formulas requires computing the inverse of the matrix $\mathbf{P}_n$. Also, when the support vector dictionary is updated, the inverse of the new kernel matrix $\mathbf{K}_{D,(n)}$ needs to be computed to calculate the ALD cost and linear dependence coefficients in (4.26) and (4.27). This matrix inversion step has computational complexity $\mathcal{O}(m^3)$ which dominates the total complexity of the algorithm. To reduce the computational complexity of the algorithm we need to find recursive update formula with lower complexity for updating $\underline{\hat{\alpha}}_n$, ρ_n and $\mathbf{K}_{D,(n)}^{-1}$.

We start by defining

$$\underline{q}_n \triangleq \mathbf{K}_{D,(n)} \underline{u}_n, \tag{4.42}$$

$$\underline{r}_n \triangleq \mathbf{P}_n^{-1} \underline{u}_n. \tag{4.43}$$

Then (4.40) and (4.41) become

$$\rho_n = \frac{1}{C}(n - \underline{q}_n^T \underline{r}_n)^{-1}, \tag{4.44}$$

$$\underline{\hat{\alpha}}_n = \underline{r}_n \rho_n. \tag{4.45}$$

Once the vectors $\underline{q}_n$ and $\underline{r}_n$ are known, ρ_n and $\underline{\hat{\alpha}}_n$ can be computed in $\mathcal{O}(m)$ time. Hence, we wish to obtain a recursive formulas for computing $\underline{q}_n$ and $\underline{r}_n$.

At time step n, based on the comparison between δ_n and γ, one of the following two cases can happen:

Case 1: $\delta_n \leq \gamma$, i.e., $\underline{\phi}(\underline{x}_n)$ is approximately linearly dependent on $\mathcal{X}_{\mathcal{D}}^{(n-1)}$ and therefore, not included in the dictionary, hence, $\mathcal{X}_{\mathcal{D}}^{(n)} = \mathcal{X}_{\mathcal{D}}^{(n-1)}$. Here the matrix $\mathbf{K}_D$ remains unchanged, i.e., $\mathbf{K}_{D,(n)} = \mathbf{K}_{D,(n-1)}$; only $\mathbf{B}$ is updated as $\mathbf{B}_n = [\mathbf{B}_{n-1}^T, \underline{\beta}_n]^T$. Then from (4.37)-(4.39), we have

$$\underline{u}_n = \underline{u}_{n-1} + \underline{\beta}_n, \tag{4.46}$$

$$\mathbf{T}_n = \mathbf{B}_{n-1}^T \mathbf{B}_{n-1} + \underline{\beta}_n \underline{\beta}_n^T \tag{4.47}$$

$$= \mathbf{T}_{n-1} + \underline{\beta}_n \underline{\beta}_n^T, \tag{4.48}$$

$$\mathbf{P}_n = \frac{\mathbf{I}_m}{C} + \left(\mathbf{T}_{n-1} + \underline{\beta}_n \underline{\beta}_n^T\right) \mathbf{K}_{D,(n)} \tag{4.49}$$

$$= \mathbf{P}_{n-1} + \underline{\beta}_n \underline{k}_n^T. \tag{4.50}$$

By applying the Sherman-Morrison formula [85] we can compute $\mathbf{P}_n^{-1}$ recursively as

$$\mathbf{P}_n^{-1} = \mathbf{P}_{n-1}^{-1} - \frac{\mathbf{P}_{n-1}^{-1} \underline{\beta}_n \underline{k}_n^T \mathbf{P}_{n-1}^{-1}}{1 + \underline{k}_{n-1}^T \mathbf{P}_n^{-1} \underline{\beta}_n}$$

$$= \mathbf{P}_{n-1}^{-1} - \underline{s}_n \underline{k}_n^T \mathbf{P}_{n-1}^{-1}, \tag{4.51}$$

where $\underline{s}_n = \frac{\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}{1+\underline{k}_n^T\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}$. Then $\underline{q}_n$ and $\underline{r}_n$ can be recursively computed as

$$\underline{q}_n = \mathbf{K}_{D,(n)}\left(\underline{u}_{n-1} + \underline{\beta}_n\right) = \underline{q}_{n-1} + \underline{k}_n, \tag{4.52}$$

$$\begin{aligned}\underline{r}_n &= \left(\mathbf{P}_{n-1}^{-1} - \underline{s}_n\underline{k}_n^T\mathbf{P}_{n-1}^{-1}\right)\left(\underline{u}_{n-1} + \underline{\beta}_n\right) \\ &= \underline{r}_{n-1} - \underline{s}_n\underline{k}_{n-1}^T\underline{r}_{n-1} + \mathbf{P}_{n-1}^{-1}\underline{\beta}_n\left(1 - \frac{\underline{k}_{n-1}^T\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}{1+\underline{k}_{n-1}^T\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}\right) \\ &= \underline{r}_{n-1} + \underline{s}_n(1 - \underline{k}_n^T\underline{r}_{n-1}).\end{aligned} \tag{4.53}$$

Finally, ρ_n and $\underline{\hat{\alpha}}_n$ are updated using (4.44)–(4.45).

Case 2: $\delta_n > \gamma$. In this case $\underline{x}_n$ is added to the dictionary, i.e., $\mathcal{X}_{\mathcal{D}}^{(n)} = \mathcal{X}_{\mathcal{D}}^{(n-1)} \cup \{\underline{x}_n\}$ and $m = m + 1$. Accordingly both $\mathbf{K}_D$ and $\mathbf{B}$ are updated as

$$\mathbf{B}_n = \begin{bmatrix}\mathbf{B}_{n-1} & \underline{0}_{n-1} \\ \underline{0}_{m-1}^T & 1\end{bmatrix}, \tag{4.54}$$

$$\mathbf{K}_{D,(n)} = \begin{bmatrix}\mathbf{K}_{D,(n-1)} & \underline{k}_n \\ \underline{k}_n^T & k_{nn}\end{bmatrix}. \tag{4.55}$$

Then from (4.37)-(4.39), we have

$$\underline{u}_n = \begin{bmatrix}\underline{u}_{n-1} \\ 1\end{bmatrix}, \tag{4.56}$$

$$\mathbf{T}_n = \begin{bmatrix}\mathbf{T}_{n-1} & \underline{0}_{m-1} \\ \underline{0}_{m-1}^T & 1\end{bmatrix}, \tag{4.57}$$

$$\mathbf{P}_n = \begin{bmatrix}\mathbf{P}_{n-1} & \mathbf{T}_{n-1}\underline{k}_n \\ \underline{k}_n^T & k_{nn} + 1/C\end{bmatrix}. \tag{4.58}$$

The recursive formulas for updating $\mathbf{K}_{D,(n)}^{-1}$ and $\mathbf{P}_n^{-1}$ can be obtained by applying the block

matrix inversion lemma [85]:

$$\mathbf{K}_{D,(n)}^{-1} = \begin{bmatrix} \mathbf{K}_{D,(n-1)}^{-1} & \underline{0}_{m-1} \\ \underline{0}_{m-1}^T & 0 \end{bmatrix} + \frac{1}{\delta_n} \begin{bmatrix} \underline{\beta}_n \\ -1 \end{bmatrix} \begin{bmatrix} \underline{\beta}_n \\ -1 \end{bmatrix}^T, \tag{4.59}$$

$$\mathbf{P}_n^{-1} = \begin{bmatrix} \mathbf{P}_{n-1}^{-1} & \underline{0}_{m-1} \\ \underline{0}_{m-1}^T & 0 \end{bmatrix} + \underline{s}_n \begin{bmatrix} -\mathbf{P}_{n-1}^{-T} \underline{k}_n \\ 1 \end{bmatrix}^T, \tag{4.60}$$

where

$$\underline{s}_n = \frac{\begin{bmatrix} -\mathbf{P}_{n-1}^{-1} \mathbf{T}_{n-1} \underline{k}_n \\ 1 \end{bmatrix}}{k_{nn} + 1/C - \underline{k}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{T}_{n-1} \underline{k}_n}. \tag{4.61}$$

Using (4.54)- (4.58) and (4.60), we obtain the recursive formulas for $\underline{q}_n$ and $\underline{r}_n$ as follows:

$$\begin{aligned} \underline{q}_n &= \begin{bmatrix} \mathbf{K}_{D,(n-1)} & \underline{k}_n \\ \underline{k}_n^T & k_{nn} \end{bmatrix} \begin{bmatrix} \underline{u}_{n-1} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \underline{q}_{n-1} + \underline{k}_n \\ \underline{k}_n^T \underline{u}_{n-1} + k_{nn} \end{bmatrix}, \end{aligned} \tag{4.62}$$

$$\begin{aligned} \underline{r}_n &= \mathbf{P}_n^{-1} \begin{bmatrix} \underline{u}_{n-1} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \underline{r}_{n-1} \\ 0 \end{bmatrix} + \underline{s}_n (1 - \underline{k}_n^T \underline{r}_{n-1}). \end{aligned} \tag{4.63}$$

Finally, ρ_n and $\underline{\hat{\alpha}}_n$ are updated using (4.44)–(4.45).

Algorithm 4.1 summarizes the sparse online least-squares one-class SVM and shows the computational complexity of each step. The overall complexity of the algorithm is $\mathcal{O}(m^2)$.

4.3 Outlier Detection

An outlier is a data point that appears different from the other data points within the sample space. In an online application scenario when a new data point becomes available, three possible cases can happen:

— it closely resembles the learned input distribution,

— it represents a change in the underlying distribution of the input data,

— it differs significantly from the learned input distribution and indicates an outlier.

The sparse online LS one-class SVM finds the hyperplane that minimizes the squared distances to the images of the training points in the feature space. Therefore, the distance from the hyperplane can be used as a measure of resemblance between a new data point and the training set. For new data $\underline{x}_n$, the distance of $\underline{\phi}(\underline{x}_n)$ from the hyperplane can be computed as

$$d(\underline{x}_n) = \frac{|\hat{\underline{\alpha}}_{n-1}^T \underline{k}_n - \rho_{n-1}|}{\sqrt{\hat{\underline{\alpha}}_{n-1}^T \mathbf{K}_{D,(n-1)} \hat{\underline{\alpha}}_{n-1}}}. \tag{4.64}$$

If $d(\underline{x}_n)$ is greater than a threshold d_{th} then we say that $\underline{x}_n$ differs significantly from the training objects seen so far, and is a possible outlier. However whether this data point indicates a change in the underlying distribution or its a true outlier still needs to be determined.

In the proposed sparse online LS one-class SVM our goal is to build a diverse support vector dictionary to approximate the input space while inducing sparsity. To achieve that we utilized ALD measure δ_n in Section 4.2. When an outlier data arrives the machine can detect a possible outlier using the distance measure. However it is difficult to distinguish between a change in underlying distribution and a true outlier. In both cases the ALD measure will be higher than the threshold. But if a true outlier is added into the support vector dictionary, it will have detrimental effect on the estimation of the decision hyperplane. On the other hand if a non-outlier but differing data point were to be excluded from the support vector dictionary we would fail to construct a diverse enough dictionary.

To overcome this we introduce another threshold γ' for the ALD measure assuming the

change in the underlying distribution is gradual. After comparing the distance threshold, before a data point is added as a support vector we have three cases:

— $\delta_n \leq \gamma$: very similar to the dictionary; only hyperplane updated.

— $\gamma < \delta_n \leq \gamma'$: somewhat similar; both dictionary and hyperplane updated.

— $\delta_n > \gamma'$: very dissimilar; discard without updating either dictionary or hyperplane.

A similar approach has been used in [53]. In an online application it is however necessary to use a time varying γ', i.e., γ' should have a large value while the size of the data set observed is still small, and then gradually decrease. This helps maintain a relatively larger dictionary at the beginning of the learning algorithm.

In power grid data, the measurement vector $\underline{x}_n$ consists of real and reactive power measurements. In the traditional state estimation, these measurements are collected from different parts of the grid and processed in a centralized manner to estimate the system states and detect outliers. In our proposed least-squares one-class SVM method, the outliers can be detected in a decentralized manner. Since the proposed algorithm does not require any information about the system states and depends entirely upon the historical observations, the outlier detection can be performed before the data is sent to a central station. By dividing the system into smaller subsystems, we can perform outlier detection in each subsystem in a distributed manner, thus reducing communication overhead. This decentralized method of outlier detection is demonstrated in the next section.

4.4 Simulation Results

To evaluate the performance of the proposed sparse online least-squares one-class SVM algorithm we applied the algorithm to a synthetic square-noise dataset [86]. For outlier detection in the power grid, we performed simulations on the IEEE 14 bus and 57 bus test systems using a Gaussian kernel $k(\underline{x}_1, \underline{x}_2) = \exp(-\|\underline{x}_1 - \underline{x}_2\|/\sigma^2)$. In this section we discuss the results of bad data detection and false data injection attack detection in the test systems, and compare performance with the largest normalized residual (r^N_{max}) test. We used the state

estimator in the MATPOWER toolbox [66] for the normalized residual test.

4.4.1 Decision boundaries on synthetic data

First we evaluate the performance of sparse online LS one-class SVM on the square-noise data set [86]. Fig. 4.1 shows the decision boundaries obtained using One-class SVM, offline LS One-class SVM and the proposed sparse online LS-OC-SVM methods. The One-class SVM decision boundary was obtained using LIBSVM-3.20 library [87] with parameters $\sigma^2 = 0.2$ and $C = 1/(0.12n)$. For both batch and online LS One-class SVM we set $\sigma^2 = 0.045$ and $C = 1$. For the online version we set $\gamma = 0.01$ and after observing 50 data points $\gamma' = \max(\exp(-m/100), 2\gamma)$ is set.

From fig. 4.1 we observe that the support vectors chosen by the standard One-class SVM algorithm are in fact outliers. This occurs because the algorithm tries to find the hyperplane such that most data points lie beyond it. Thus the data points located at the edge of the data set are chosen as support vectors. On the other hand, the offline LS One-class SVM obtains a decision boundary that is more representative of the data. The drawback is that now every data point acts as a support vector. Finally the decision region obtained by the proposed method is smoother and does not pick any outlier as a support vector.

4.4.2 Bad data detection

We next implemented the proposed algorithm for bad data detection in the IEEE 14 bus test system [65]. Fig. 4.2 shows the network diagram of the test system. For distributed bad data detection using least-squares one-class SVM, we divide the 14 bus system into two subsystems as shown in Fig. 4.2. For our simulations, we only consider bad data detection in the region enclosed in blue lines. The measurement configuration for state estimation and the normalized residual test is given in Table 4.1. Under this measurement configuration P_{4-7} is a critical measurement. The residual for this measurement will always be zero and

therefore, any bad data in this measurement cannot be identified by the r^N_{max} test.

In the simulations, we assume that the system is operating in a quasi steady state. We created a data stream of 1500 samples by adding 3% measurement noise to the true measurements. For outlier detection using least-squares one-class SVM, we consider the measurement vector $\underline{X} = [P_1,\ P_2,\ P_3,\ P_{1-2},\ P_{1-5},\ P_{4-7},\ P_{4-2},\ P_{4-5},\ P_{4-9},\ P_{2-5},\ P_{5-6}]^T$. To study the efficacy of the proposed method in detecting bad data in critical measurement P_{4-7}, we randomly injected 50 bad data into the data stream with gross errors of magnitudes $25 - 50\%$ of the true measurement. For multiple interacting bad data detection, we injected bad data in the interacting pair P_1 and P_{1-2}. The parameters used for online LS-OC-SVM are $C = 8$, $\sigma^2 = 0.011$, $\gamma = 0.0002$, $\gamma' = \max(\exp(-m/5), 2\gamma)$. Fig. 4.3 shows the comparison of the detection rates of r^N_{max} test and least-squares one-class SVM method, averaged over 100 simulations. We observe that r^N_{max} test fails to detect any bad data in a critical measurement and has low detection rate for multiple interacting bad data. In contrast, the proposed online algorithm has high positive detection rate ($> 95\%$) in both critical and multiple interacting bad data while achieving a low false detection rate.

Next, we tested the proposed online algorithm for bad data detection in the IEEE 57 bus system [65]. Fig. 4.4 shows the schematic diagram of a section of the test system with the subsystem under consideration enclosed in blue lines. We again created a data stream of 1500 data points for online bad data detection in critical and multiple interacting measurements. We considered the measurement vector $\underline{X} = [P_{38-37},\ P_{38-22},\ P_{38-44},\ P_{38-49},\ P_{38-48},\ P_{13-49},\ P_{47-46},\ P_{39-57},\ P_{22-21}]^T$. We randomly injected 50 bad measurements in the critical measurement P_{38-37} and interacting pair P_{38-44} and P_{13-49}, with gross error magnitudes in the range of 25% to 50% of the measured values. The parameters used for the sparse online least-squares one-class SVM are $C = 16$, $\sigma^2 = 0.012$, $\gamma = 0.0002$, $\gamma' = \max(\exp(-m/5), 2\gamma)$. Fig. 4.5 shows the comparison of the detection rates of r^N_{max} test and least-squares one-class SVM method, averaged over 100 simulations. We observe again that the proposed method significantly outperforms the r^N_{max} test in both critical and multiple interacting bad data detection.

Table 4.2 summarizes the results of critical and multiple bad data detection in the IEEE 14 bus and 57 bus test systems, showing the means and standard deviations of the detection rates. In every test scenario, the proposed algorithm showed significant improvement over the r^N_{max} test. Our data driven one-class SVM also has the advantage over the normalized residual test in that we detect bad data in a distributed manner without requiring multiple runs of the computationally expensive state estimator.

4.4.3 False data injection attack detection

To study the feasibility of the proposed least-squares one-class SVM method for detecting malicious data attacks, we performed simulations on the IEEE 14 bus system [65]. Data injection attacks can be designed to be undetectable by the r^N_{max} test [10, 35]. For instance, an attacker can pass the r^N_{max} test if the attacker has knowledge of the system structure ($\underline{h}$) and can manipulate multiple measurements at the same time. In AC state estimation the attacker also needs to have an estimate of the system state [35]. If an attacker designed an attack vector as $\underline{a} = \underline{h}(\hat{\underline{x}} + \underline{c}) - \underline{h}(\hat{\underline{x}})$ and changed the measurement to $\underline{z}_{bad} = \underline{z} + \underline{a}$, then $\underline{a}$ can pass the bad data detection test. The AC state estimator will yield an erroneous state $\underline{x}_{bad} = \hat{\underline{x}} + \underline{c}$. Two types of data attacks are usually studied in the literature: attack on state variable(s) and attack on certain measurements [35, 38]. In our simulations we study the attacks targeting state variable(s).

We investigate attacks targeting one state variable from the set $\mathcal{A} = \{V_2, V_3, V_4, V_5, \theta_2, \theta_3, \theta_4, \theta_5\}$. To successfully alter any of the state variables without being detected, the attack must change all the measurements that depend upon that state variable. Once it has been determined which measurements need to be altered, the real and reactive power flows from

bus i to j can be calculated from [4]

$$\begin{aligned} P_{ij} = V_i^2(g_{si} + g_{ij}) - V_i V_j g_{ij} \cos(\theta_i - \theta_j) \\ - V_i V_j b_{ij} \sin(\theta_i - \theta_j), \end{aligned} \tag{4.65}$$

$$\begin{aligned} Q_{ij} = -V_i^2(b_{si} + b_{ij}) - V_i V_j g_{ij} \sin(\theta_i - \theta_j) \\ + V_i V_j b_{ij} \cos(\theta_i - \theta_j), \end{aligned} \tag{4.66}$$

where g_{si}, b_{si}, g_{ij} and b_{ij} are network parameters. The power injected at bus i is then

$$P_i = \sum_j P_{ij}, \tag{4.67}$$

$$Q_i = \sum_j Q_{ij}. \tag{4.68}$$

To simulate attack on a state variable in $\mathcal{A}$, we generate a data stream of 1500 observations where the last 100 observations are malicious data containing measurements to alter the state variable by $10 - 20\%$ of its true value. We repeat this procedure for each of the state variables in $\mathcal{A}$. Fig. 4.6 shows the detection rate of the proposed online least-squares one-class SVM compared with the r^N_{max} test, averaged over 100 simulations for each variable. As expected from the attack design, the r^N_{max} test fails to detect any of the attacks. On the other hand, our algorithm achieved a 100% true positive detection rate in detection of the false data injection attacks while maintaining $< 0.005\%$ average false detection rate.

Algorithm 4.1: SPARSE ONLINE LS ONE-CLASS SVM

Initialize: $\mathcal{X}_D^{(1)} = \{\underline{x}_1\}$, $m = 1$, $\mathbf{K}_{\mathcal{D},(1)} = [k_{11}]$, $\mathbf{K}_{D,(1)}^{-1} = [1/k_{11}]$, $\mathbf{B}_1 = [1]$, $P_1^{-1} = (1/C + k_{11})^{-1}$, $\hat{\underline{\alpha}}_1 = 1$, $\rho_1 = (1 + Ck_{11})/C$, $\underline{q}_1 = [k_{11}]$, $\underline{r}_1 = (1/C + k_{11})^{-1}$, $\mathbf{T}_1 = 1$, $\underline{u}_1 = 1$.

for $n = 2, 3, \ldots$ **do**

compute $\underline{k}_n$	$\mathcal{O}(m)$
$\underline{\beta}_n = \mathbf{K}_{D,(n-1)}^{-1}\underline{k}_n$	$\mathcal{O}(m^2)$
$\delta_n = k_{nn} - \underline{k}_n^T\underline{\beta}_n$	$\mathcal{O}(m)$
$d(\underline{x}_n) = \frac{\lvert\hat{\underline{\alpha}}_{n-1}^T\underline{k}_n - \rho_{n-1}\rvert}{\sqrt{\hat{\underline{\alpha}}_{n-1}^T\mathbf{K}_{D,(n-1)}\hat{\underline{\alpha}}_{n-1}}}$	
if $d(\underline{x}_n) > d_{th}$ ***and*** $\delta_n > \gamma'$ **then**	
declare outlier	
else	
if $\delta_n \leq \gamma$ **then** `// dictionary unchanged`	
$\mathcal{X}_D^{(n)} = \mathcal{X}_D^{(n-1)}$	
$\mathbf{K}_{D,(n)}^{-1} = \mathbf{K}_{D,(n-1)}^{-1}$	
$\mathbf{B}_n = \begin{bmatrix}\mathbf{B}_{n-1}^T & \underline{\beta}_n\end{bmatrix}^T$	
Compute $\mathbf{T}_n = \mathbf{T}_{n-1} + \underline{\beta}_n\underline{\beta}_n^T$	$\mathcal{O}(m^2)$
Compute $\underline{u}_n = \underline{u}_{n-1} + \underline{\beta}_n$	$\mathcal{O}(m)$
Compute $\underline{s}_n = \frac{\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}{1+\underline{k}_n^T\mathbf{P}_{n-1}^{-1}\underline{\beta}_n}$	$\mathcal{O}(m^2)$
Compute $\mathbf{P}_n^{-1}$ using (4.51)	$\mathcal{O}(m^2)$
Compute $\underline{q}_n$ using (4.52)	$\mathcal{O}(m)$
Compute $\underline{r}_n$ using (4.53)	$\mathcal{O}(m)$
else `// add` $\underline{x}_n$ `to dictionary`	
$\mathcal{X}_D^{(n)} = \mathcal{X}_D^{(n-1)} \cup \{\underline{x}_n\}$	
$m = m + 1$	
$\mathbf{B}_n = \text{blkdiag}(\mathbf{B}_{n-1}, 1)$	
$\mathbf{T}_n = \text{blkdiag}(\mathbf{T}_{n-1}, 1)$	
$\underline{u}_n = \begin{bmatrix}\underline{u}_{n-1}^T & 1\end{bmatrix}^T$	
Compute $\mathbf{K}_{D,(n)}^{-1}$ using (4.59)	$\mathcal{O}(m^2)$
Compute $\underline{s}_n$ using (4.61)	$\mathcal{O}(m^2)$
Compute $\mathbf{P}_n^{-1}$ using (4.60)	$\mathcal{O}(m^2)$
Compute $\underline{q}_n$ using (4.62)	$\mathcal{O}(m)$
Compute $\underline{r}_n$ using (4.63)	$\mathcal{O}(m)$
end	
update ρ_n and $\hat{\underline{\alpha}}_n$ using (4.44) and (4.45)	$\mathcal{O}(m)$
end	

end

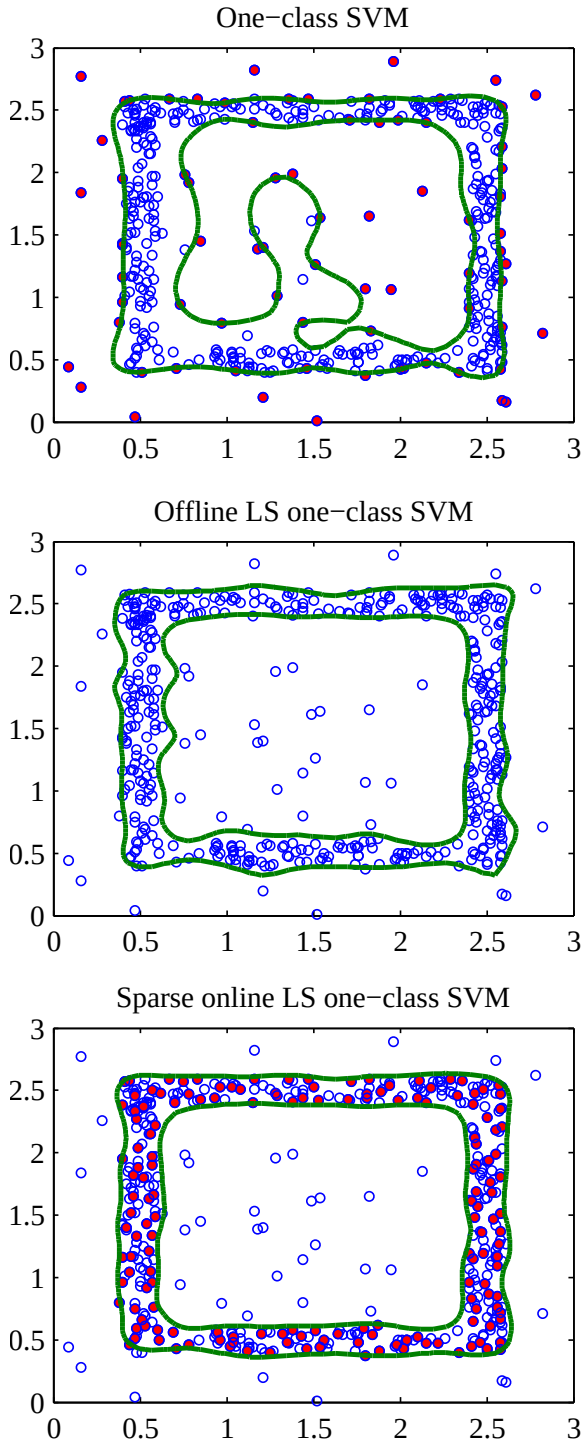


Figure 4.1: Comparison of decision boundaries obtained by OC-SVM ($\sigma = 0.2$, $C = 1/(0.12n)$), offline LS-OC-SVM ($\sigma^2 = 0.045$, $C = 1$) and sparse online LS-OC-SVM ($\sigma^2 = 0.045$, $C = 1$, $\gamma_1 = 0.01$). The solid red circles represent the support vectors in OC-SVM and sparse online LS-OC-SVM.

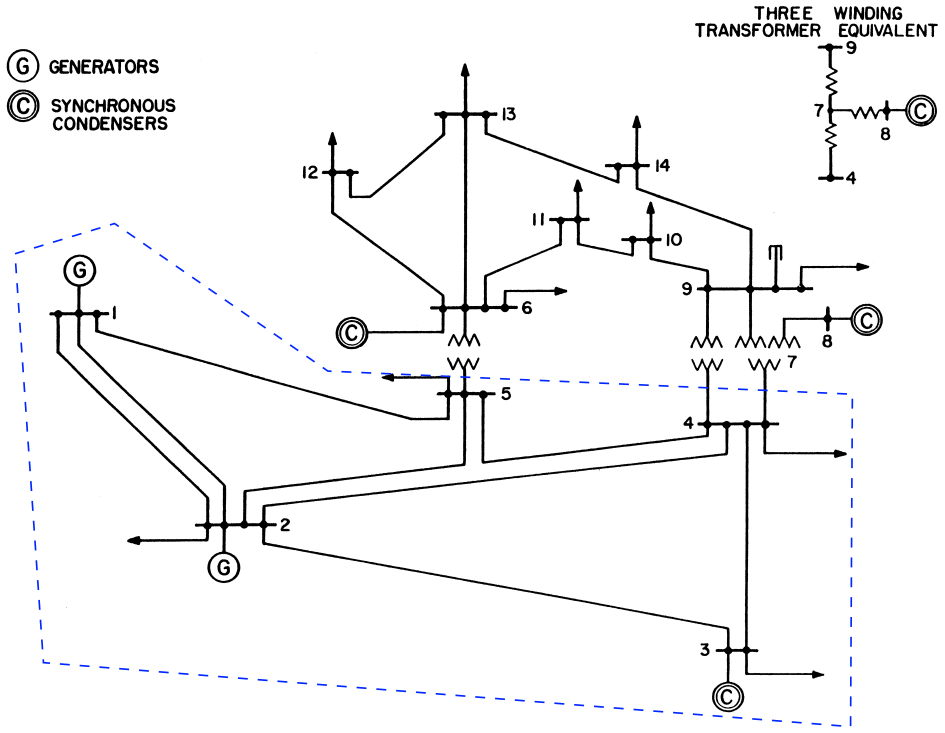


Figure 4.2: IEEE 14 bus test system [65]. The blue line shows the area under consideration.

Table 4.1: Measurement configuration for IEEE 14 bus test system

Power injection	At buses 1, 2, 3, 8
Power flow	At branches 1-2, 1-5, 2-4, 2-5, 4-5, 4-7, 4-9, 5-6, 6-11, 6-13, 9-10, 9-14, 12-13
Voltage magnitudes	At buses 3, 6, 8, 10, 14

Table 4.2: Performances of r^N_{max} test and proposed LS-OC-SVM method in detecting bad data in critical and multiple interacting measurements

		r^N_{max} test accuracy (%)		LS-OC-SVM accuracy (%)	
		True positive	False positive	True positive	False positive
Critical	14 bus	0 ± 0	0 ± 0	95.04 ± 3.12	0.74 ± 1.2
	57 bus	0 ± 0	0 ± 0	93.20 ± 4.49	5.4 ± 3.09
Multiple interacting	14 bus	46.28 ± 6.73	0 ± 0	95.16 ± 3.23	1.64 ± 1.76
	57 bus	70.92 ± 6.22	0 ± 0	94.04 ± 4.04	5.42 ± 3.24

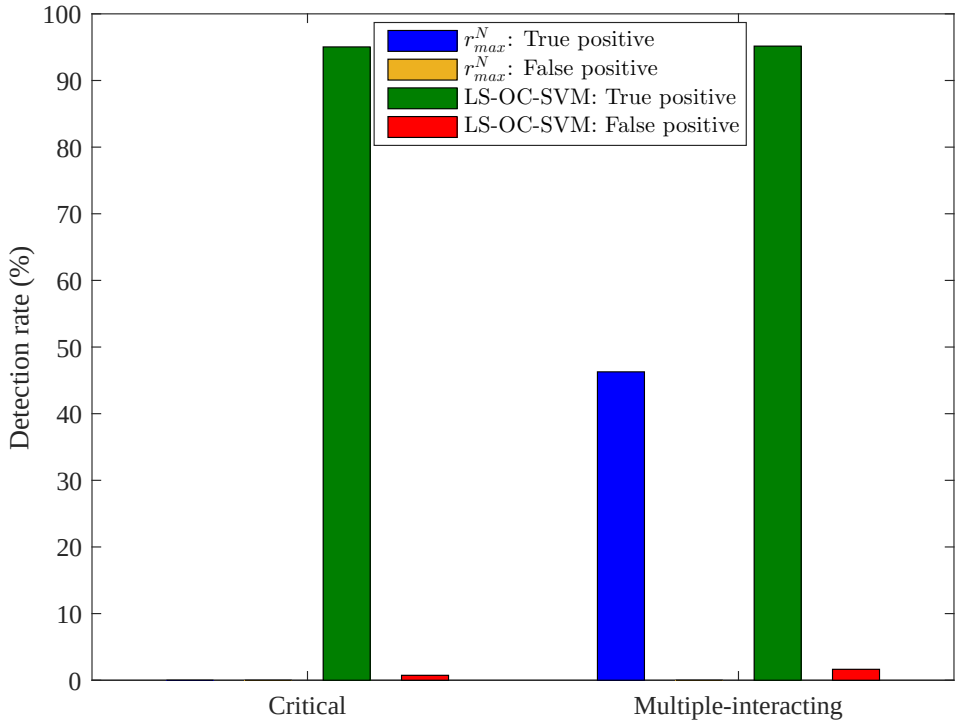


Figure 4.3: Bad data detection rates in IEEE 14 bus test system

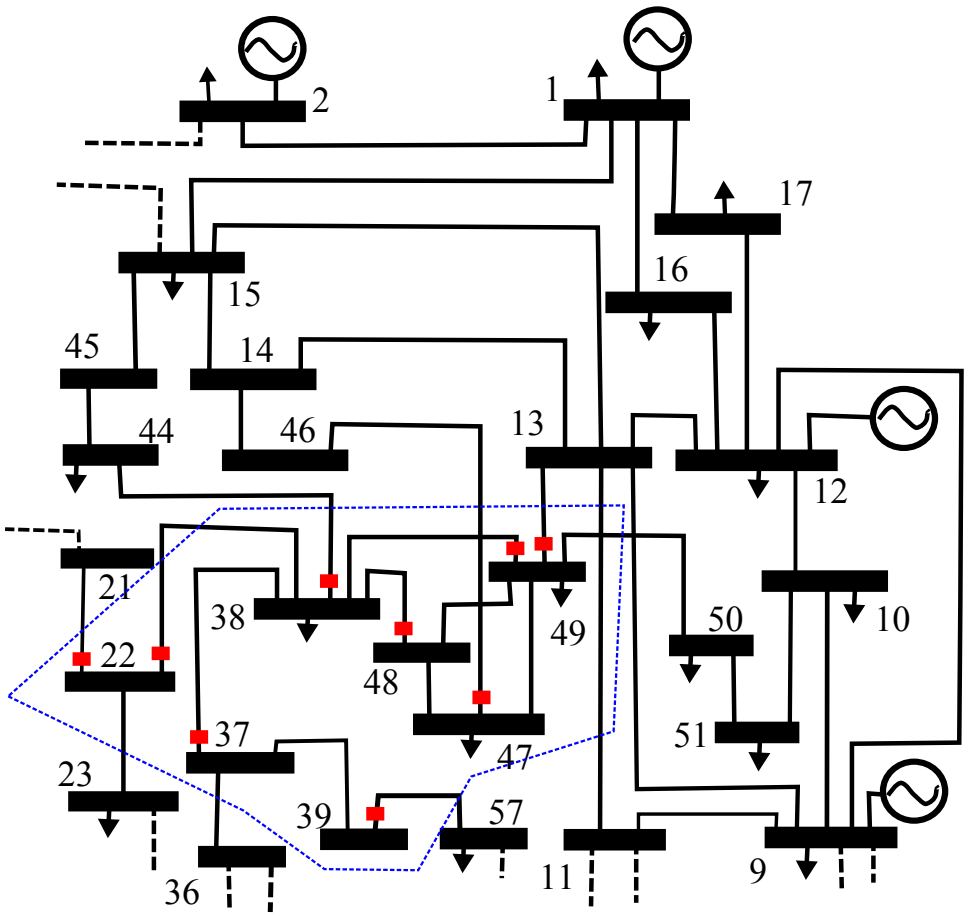


Figure 4.4: IEEE 57 bus test system [65]. The blue line shows the area under consideration and the red squares are the available power flow measurements.

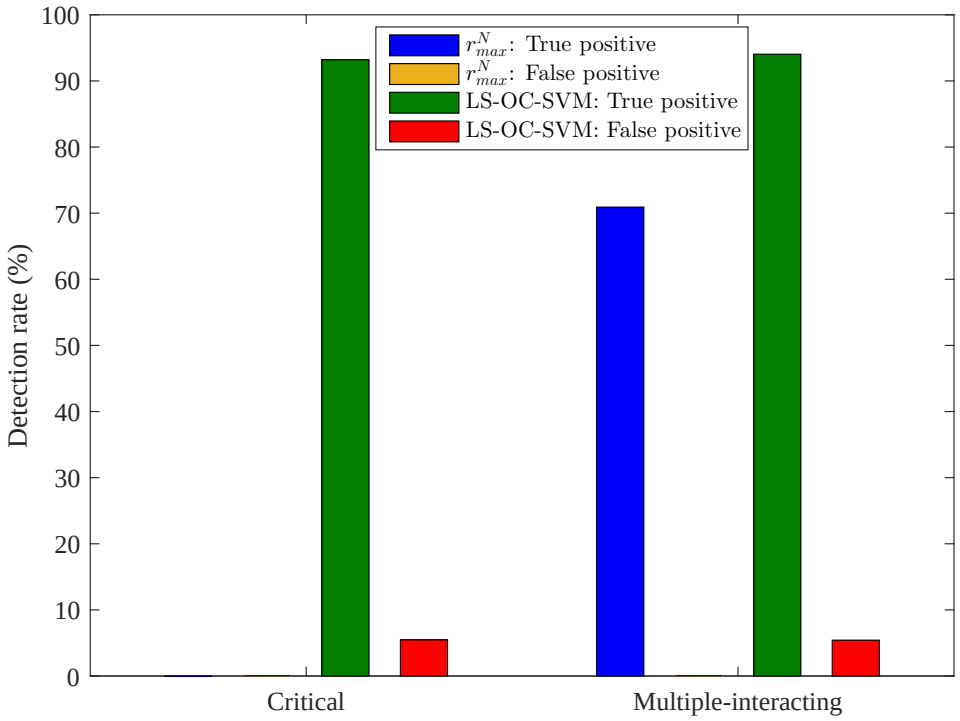


Figure 4.5: Bad data detection rates in IEEE 57 bus test system

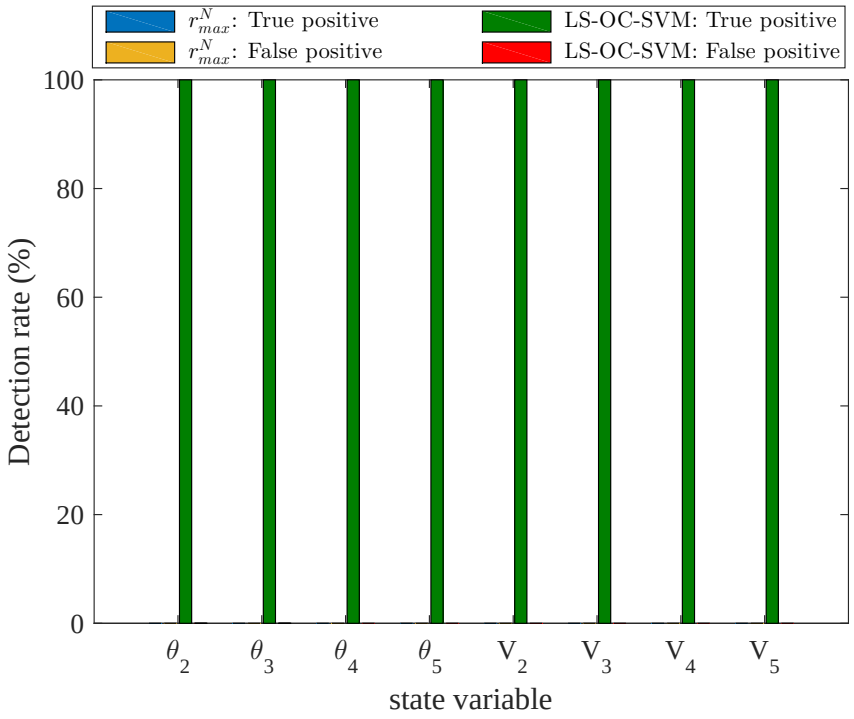


Figure 4.6: Detection rate of false data injection attacks on different state variables in IEEE 14 bus system

5

Conclusion

In this thesis we investigated the performance of sensor placement algorithms and quality of sensor data. The first part of the thesis considers the optimal placement of m sensors among n locations. The optimal solution involves solving an integer programming problem and is NP-complete [5]. We came up with a series of approximation algorithms, including a greedy algorithm and a dynamic programming based algorithm, and their variations that have polynomial complexities. Through simulations we verified that their performances are close to the optimal solution. To further understand the performance of the optimal solution we come up with a series of nested upper bounds (using matrix pencils, generalized eigenvectors, and matrix manipulations) that give tighter upper bounds at increasing complexity. From these nested upper bounds we found a new set of approximation algorithms that are based

on finding projections of unit binary vectors on the subspace generated by the nested upper bounds.

In the second part we examined two outlier detection algorithms to ensure data quality in the power grid. The proposed algorithms identify outliers in an online fashion *before* the state estimation process, without relying on the *observability* of the grid. The online kernel density estimation based outlier detection algorithm treats the bad measurements as spatial and temporal outliers in the network and identifies these in a decentralized manner without depending on the observability of the grid. From the simulations performed on the IEEE 14-bus system we observed that the proposed method outperforms the largest normalized residual test. To overcome the high computational cost of the KDE based outlier detector, we then proposed a sparse online least-squares one-class SVM for outlier detection in the power grid. The algorithm has polynomial computational complexity, and the online and decentralized properties of this method provides remarkable advantages over the normalized residual test. Through simulations on the IEEE benchmark test systems we have verified the efficacy of the proposed method for detecting bad data in critical and multiple interacting measurements, as well as malicious data injection attacks. Both of the proposed methods detect outliers in a distributed manner without requiring global or local estimation of the system states.

5.1 Future Directions

5.1.1 Sensor Placement in Power Grid

There are many possible future directions for this research. We would like to come up with tighter upper bounds for the optimal performance. Through simulations we showed that many of the approximation algorithms perform quite well when compared to the optimal algorithm. It would be interesting to come up with tighter theoretical bounds. Algorithms can also be considered when there are additional constraints to the problem (e.g. maximize

observability), for incremental cases (e.g. want to add some sensors given some sensors have already been placed which is a generalization of the greedy algorithm), and for dynamic state estimation problems (important for state estimation for the electrical power grid). Another possible future direction is the dynamic sensor *selection* problem. In this research, we considered the sensors locations to be static. An interesting direction of research would be designing and analyzing performance of algorithms when a subset of sensors are chosen dynamically from a larger set of deployed sensors. The motivation for this line of research comes from the minimization of communication requirement in the grid.

In the simulation section we had an example for an IEEE 57-bus test system. Approximation algorithms perform well and we could apply these algorithms for placement of meters at the distribution level for microgrid state estimation problems. The microgrid has specific local and hierarchical correlations that can be represented by graphs that for the most part are radial. In these cases perhaps distributed algorithms (such as *belief propagation*) can be used to find good approximate solutions to the sensor placement problem.

5.1.2 Online Outlier Detection

In our future work we would like to address the problem of identification of bad measurements using least-squares one-class SVM, along with detection. Another possible avenue of research is the incorporation of the dynamic behavior of the power grid by considering a non-stationary learning model. If the data generating distribution changes with time it becomes necessary to remove the older support vectors to better estimate the changing support of the distribution. Extending the online least-squares one-class SVM algorithm to utilize multiple kernels is yet another possible avenue of research. With more and more types data sources being deployed, the idea of utilizing the data from heterogeneous sources in the analysis promises potential improvements in the decision making process.

In false data injection attack detection, we considered attacks targeting a single state variable. In our future work, we would also like to investigate other types of data attacks, e.g., replay

attacks, load redistribution attacks etc. Yet another possible direction of future research is the theoretical analysis of the change in the underlying distribution during false data injection attacks. A theoretical analysis of the attacks vectors may help determine the thresholds for outlier detection.

A

Appendix

A.1 Proof of Theorem 2.1

Proof. Using (2.21), for any $\mathbf{F} \in \mathcal{F}^{[m\times n]}$, we have

$$J_{\langle \mathbf{A},\, \mathbf{B}\rangle}(\mathbf{F}) = \operatorname{tr}\left\{ \left[\mathbf{F}\mathbf{B}\mathbf{F}^T\right]^{-1} \mathbf{F}\mathbf{A}\mathbf{F}^T \right\}. \tag{A.1}$$

Let

$$\mathbf{G} = \mathbf{F}\mathbf{B}\mathbf{F}^T. \tag{A.2}$$

Since $\mathbf{G}$ is symmetric positive definite, there exists an $m \times m$ invertible matrix $\mathbf{G}^{1/2}$ such that

$$\mathbf{G} = \mathbf{G}^{1/2}\mathbf{G}^{T/2}. \tag{A.3}$$

Define $\mathbf{M}$ of size $m \times n$ as

$$\mathbf{M} = \mathbf{G}^{-1/2}\mathbf{F}\mathbf{V}^{-T}. \tag{A.4}$$

Combining Lemma A with (A.3) - (A.4) we have

$$\mathbf{M}\mathbf{M}^T = \mathbf{I}_m. \tag{A.5}$$

Further applying Lemma A and (A.2) - (A.4), the efficacy now reads

$$\begin{aligned} J_{\langle \mathbf{A}, \mathbf{B} \rangle}(\mathbf{F}) &= \operatorname{tr}\left[\mathbf{G}^{-1}\mathbf{F}\mathbf{A}\mathbf{F}^T\right] && \text{(A.6)} \\ &= \operatorname{tr}\left[\mathbf{G}^{-1}\mathbf{F}\mathbf{V}^{-T}\mathbf{D}\mathbf{V}^{-1}\mathbf{F}^T\right] && \text{(A.7)} \\ &= \operatorname{tr}\left[\mathbf{G}^{-1}\mathbf{G}^{1/2}\mathbf{M}\mathbf{D}\mathbf{M}^T\mathbf{G}^{T/2}\right] && \text{(A.8)} \\ &= \operatorname{tr}\left[\mathbf{M}\mathbf{D}\mathbf{M}^T\right]. && \text{(A.9)} \end{aligned}$$

Under the constraint (A.5), the efficacy in (A.6) is maximized when $\mathbf{M}$ consists of the m eigenvectors of $\mathbf{D}$ that correspond to the m highest eigenvalues. Since $\mathbf{D}$ is diagonal, eigenvectors of $\mathbf{D}$ are the natural basis vectors. Therefore,

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix}. \tag{A.10}$$

Thus, by substituting (A.10) into (A.6), the optimal efficacy $J^*_{\langle \mathbf{A}, \mathbf{B} \rangle}$ becomes

$$J^*_{\langle \mathbf{A}, \mathbf{B} \rangle} = \operatorname{tr}\left\{ \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{D} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix}^T \right\} = \sum_{j=1}^{m} \delta_j. \tag{A.11}$$

Now set $\mathbf{F} = \mathbf{M}^*\mathbf{V}^T = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{V}^T$, and verify (using Lemma A) that this choice of $\mathbf{F}$ has

efficacy $J_{\langle \mathbf{A},\mathbf{B} \rangle}(\mathbf{F})$ that equals to the maximal efficacy in (A.11) . Consequently, the optimal argument must equal

$$\mathbf{F}^*_{\langle \mathbf{A},\mathbf{B} \rangle} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \end{bmatrix} \mathbf{V}^T = \begin{bmatrix} \underline{v}_1 & \cdots & \underline{v}_m \end{bmatrix}^T. \tag{A.12}$$

■

A.2 Proof of Lemma B

Proof. For any $\mathbf{F} \in \mathcal{F}^{[(m-k)\times(n-k)]}$, using (2.21) we express the efficacy as

$$J_{\langle \mathbf{A},\mathbf{B} \rangle}\left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) = \mathrm{tr}\left\{ \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \mathbf{B} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix}^T \right)^{-1} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix}^T \right\}. \tag{A.13}$$

Using (2.40), (2.41) and (2.42), we express $\mathbf{B}$ in terms of $\mathbf{P}_k$, $\mathbf{Q}_k$ and $\mathbf{R}_k$ as

$$\mathbf{B} = \begin{bmatrix} \mathbf{P}_k & \mathbf{Q}_k \\ \mathbf{Q}_k^T & \mathbf{R}_k \end{bmatrix}. \tag{A.14}$$

Substituting $\mathbf{B}$ into (A.13) and using the partitioned matrix inversion lemma [88], we write (A.13) as

$$\begin{aligned} J_{\langle \mathbf{A},\mathbf{B} \rangle}\left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) &= \mathrm{tr}\left\{ \mathbf{A} \begin{bmatrix} \mathbf{P}_k^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right\} \\ &+ \mathrm{tr}\left\{ \begin{bmatrix} \mathbf{P}_k^{-1}\mathbf{Q}_k\mathbf{F}^T \\ -\mathbf{I}_{m-k} \end{bmatrix} \left\{ \mathbf{F}\left(\mathbf{R}_k - \mathbf{Q}^T\mathbf{P}_k^{-1}\mathbf{Q}_k \right)\mathbf{F}^T \right\}^{-1} \begin{bmatrix} \mathbf{P}_k^{-1}\mathbf{Q}_k\mathbf{F}^T \\ -\mathbf{I}_{m-k} \end{bmatrix}^T \right. \\ &\left. \times \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix}^T \right\}. \end{aligned} \tag{A.15}$$

Using (2.37), (2.38) and (2.39), we simplify (A.15) as

$$J_{\langle \mathbf{A}, \mathbf{B} \rangle} \left(\begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{bmatrix} \right) = t_k + \operatorname{tr} \left\{ \left(\mathbf{F} \mathbf{B}_k \mathbf{F}^T \right)^{-1} \mathbf{F} \mathbf{A}_k \mathbf{F}^T \right\} \tag{A.16}$$

$$= t_k + J_{\langle \mathbf{A}_k, \mathbf{B}_k \rangle}(\mathbf{F}). \tag{A.17}$$

■

Bibliography

[1] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: theory, efficient algorithms, and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[2] Y.-F. Huang, S. Werner, J. Huang, N. Kashyap, and V. Gupta, "State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid," *Signal Processing Magazine, IEEE*, vol. 29, no. 5, pp. 33–43, 2012.

[3] A. G. Phadke and J. S. Thorp, *Synchronized Phasor Measurements and Their Applications*. Springer, 2008.

[4] A. Abur and A. G. Expósito, *Power System State Estimation: Theory and Implementation*. New York, NY: Marcel Dekker, Inc., 2004.

[5] D. Brueni and L. Heath, "The PMU placement problem," *SIAM Journal on Discrete Mathematics*, vol. 19, no. 3, pp. 744–761, Dec. 2005.

[6] Q. Li, R. Negi, and M. D. Ilić, "Phasor measurement units placement for power system state estimation: A greedy approach," in *PES General Meeting, 2011 IEEE*, Detroit, MI, July 2011, pp. 1–8.

[7] V. Kekatos, G. B. Giannakis, and B. Wollenberg, "Optimal placement of phasor mea-

surement units via convex relaxation," *Power Systems, IEEE Transactions on*, vol. 27, no. 3, pp. 1521–1530, Aug. 2012.

[8] C. C. Aggarwal, *Outlier analysis*. Springer Science & Business Media, 2013.

[9] A. Monticelli, "Electric power system state estimation," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 262–282, 2000.

[10] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proc. 16th ACM Conf. Computer and Communications Security*, 2009, pp. 21–32.

[11] J. Zhao, G. Zhang, K. Das, G. N. Korres, N. M. Manousakis, A. K. Sinha, and Z. He, "Power system real-time monitoring by using pmu-based robust state estimation method," *IEEE Trans. Smart Grid*, vol. 7, no. 1, pp. 300–309, Jan 2016.

[12] A. S. Dobakhshari and A. M. Ranjbar, "A wide-area scheme for power system fault location incorporating bad data detection," *IEEE Trans. Power Delivery*, vol. 30, no. 2, pp. 800–808, April 2015.

[13] M. Brown Do Coutto Filho, J. C. Stacchini de Souza, and M. A. Ribeiro Guimaraens, "Enhanced bad data processing by phasor-aided state estimation," *Power Systems, IEEE Transactions on*, vol. 29, no. 5, pp. 2200–2209, Sept 2014.

[14] S. S. Dhillon, K. Chakrabarty, and S. S. Iyengar, "Sensor placement for grid coverage under imprecise detections," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2, Jul. 2002, pp. 1581–1587.

[15] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3, New Orleans, LA, 2003, pp. 1609–1614.

[16] X. Liu and P. Mahapatra, "On the deployment of wireless sensor nodes," in *Proc. of the*

3rd Int. Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks, San Diego, CA, Jul. 2005.

[17] D. Bajovic, B. Sinopoli, and J. Xavier, "Sensor selection for event detection in wireless sensor networks," *Signal Processing, IEEE Transactions on*, vol. 59, no. 10, pp. 4938–4953, 2011.

[18] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Efficient sensor position selection using graph signal sampling theory," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 6225–6229.

[19] J. Chen and A. Abur, "Placement of PMUs to enable bad data detection in state estimation," *Power Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 1608–1615, 2006.

[20] B. Xu and A. Abur, "Observability analysis and measurement placement for systems with PMUs," in *Power Systems Conference and Exposition, 2004. IEEE PES*, vol. 2, New York, NY, Oct. 2004, pp. 943–946.

[21] M. Nazari-Heris and B. Mohammadi-Ivatloo, "Application of heuristic algorithms to optimal PMU placement in electric power systems: An updated review," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 214 – 228, 2015.

[22] N. M. Manousakis, G. N. Korres, and P. S. Georgilakis, "Taxonomy of PMU placement methodologies," *IEEE Trans. Power Syst.*, vol. 27, no. 2, pp. 1070–1077, May 2012.

[23] D. Dua, S. Dambhare, R. K. Gajbhiye, and S. A. Soman, "Optimal multistage scheduling of PMU placement: an ilp approach," *IEEE Trans. Power Del.*, vol. 23, pp. 1812–1820, Oct. 2008.

[24] S. Chakrabarti, G. K. Venayagamoorthy, and E. Kyriakides, "PMU placement for power system observability using binary particle swarm optimization," in *Power Engineering Conference, 2008. AUPEC '08. Australasian Universities*, Sydney, Australia, Dec. 2008, pp. 1–5.

[25] V. Kekatos and G. B. Giannakis, "A convex relaxation approach to optimal placement of phasor measurement units," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on*, San Juan, Puerto Rico, Dec. 2011, pp. 145–148.

[26] F. Aminifar, A. Khodaei, M. Fotuhi-Firuzabad, and M. Shahidehpour, "Contingency-constrained pmu placement in power networks," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 516–523, Feb 2010.

[27] C.-S. Liao, T.-J. Hsieh, X.-C. Guo, J.-H. Liu, and C.-C. Chu, "Hybrid search for the optimal PMU placement problem on a power grid," *European Journal of Operational Research*, vol. 243, no. 3, pp. 985 – 994, 2015.

[28] Y. Zhao, P. Yuan, Q. Ai, and T. Lv, "Optimal PMU placement considering topology constraints," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 240 – 248, 2015.

[29] H. Moradi Koupaie, S. Ibrahim, and J. Hosseinkhani, "Outlier detection in stream data by clustering method," *International J. Advanced Computer Science and Inf. Technology*, vol. 2, no. 3, pp. 25–34, 2013.

[30] J. Khwanram and P. Damrongkulkamjorn, "Multiple bad data identification in power system state estimation using particle swarm optimization," in *ECTI-CON 2009*, May 2009, pp. 2–5.

[31] H.-M. Jeong, J. H. Park, and H.-S. Lee, "Multiple bad data identification using binary particle swarm optimization," *Journal of International Council on Electrical Engineering*, vol. 1, no. 3, pp. 269–273, 2011.

[32] E. N. Asada, A. V. Garcia, and R. Romero, "Identifying multiple interacting bad data in power system state estimation," in *IEEE PES General Meeting, 2005*, June 2005, pp. 571–577.

[33] M. A. Rahman and H. Mohsenian-Rad, "False data injection attacks with incomplete in-

formation against smart power grids," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 3153–3158.

[34] Z. H. Yu and W. L. Chin, "Blind false data injection attack using pca approximation method in smart grid," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1219–1226, May 2015.

[35] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sept 2012.

[36] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 645–658, Dec 2011.

[37] M. Esmalifalak, L. Liu, N. K. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–9, 2014.

[38] G. Chaojun, P. Jirutitijaroen, and M. Motani, "Detecting false data injection attacks in ac state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2476–2483, Sept 2015.

[39] S. Li, Y. Yilmaz, and X. Wang, "Quickest detection of false data injection attack in wide-area smart grids," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2725–2735, Nov 2015.

[40] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Networks and Learning Systems*, vol. PP, no. 99, 2015.

[41] H. Sedghi and E. Jonckheere, "Statistical structure learning to ensure data integrity in smart grid," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1924–1933, July 2015.

[42] A. Giani, E. Bitar, M. Garcia, M. McQueen, P. Khargonekar, and K. Poolla, "Smart

grid data integrity attacks," *Smart Grid, IEEE Transactions on*, vol. 4, no. 3, pp. 1244–1253, Sept 2013.

[43] M. Uddin, A. Kuh, A. Kavcic, and T. Tanaka, "Approximate solutions and performance bounds for the sensor placement problem," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, Tainan, Taiwan, Nov. 2012, pp. 31–36.

[44] M. Uddin, A. Kuh, A. Kavcic, T. Tanaka, and D. P. Mandic, "Grid monitoring: Bounds on performances of sensor placement algorithms," in *ENERGY 2013, The Third Int. Conf. on Smart Grids, Green Communications and IT Energy-aware Technologies*, Lisbon, Portugal, Mar. 2013, pp. 89–95.

[45] M. Uddin, A. Kuh, and A. Kavcic, "Nested bounds for the constrained sensor placement problem," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 4216–4220.

[46] M. S. Uddin, A. Kuh, A. Kavcic, and T. Tanaka, "Nested performance bounds and approximate solutions for the sensor placement problem," *APSIPA Transactions on Signal and Information Processing*, vol. 3, Mar 2014. [Online]. Available: https://www.cambridge.org/core/article/nested-performance-bounds-and-approximate-solutions-for-the-sensor-placement-problem/F391F848E287F43A23700229986DAD8C

[47] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[48] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[49] Y.-S. Choi, "Least squares one-class support vector machine," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1236 – 1240, 2009.

[50] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. River Edge, NJ: World Scientific, 2002.

[51] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[52] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[53] W. Liu, I. Park, and J. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Networks*, vol. 20, no. 12, pp. 1950–1961, Dec 2009.

[54] T. Wang, J. Chen, Y. Zhou, and H. Snoussi, "Online least squares one-class support vector machines-based abnormal visual event detection," *Sensors*, vol. 13, no. 12, pp. 17 130–17 155, 2013.

[55] M. S. Uddin, A. Kuh, Y. Weng, and M. D. Ilić, "Online bad data detection using kernel density estimation," in *PES General Meeting, 2015 IEEE*, Denver, CO, July 2015.

[56] M. S. Uddin and A. Kuh, "Online least-squares one-class support vector machine for outlier detection in power grid data," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 2628–2632.

[57] ——, "Sparse online least squares one-class support vector machine for outlier detection in power grid," *IEEE Transaction on Smart Grid*, submitted for publication.

[58] A. Bergen and V. Vittal, *Power Systems Analysis*. Pearson/Prentice Hall, 2000.

[59] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: PTR Prentice Hall, 1993.

[60] M. L. Crow, *Computational Methods for Electric Power Systems*, 2nd ed. New York, NY: CRC Press, 2009.

[61] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[62] J. Forney, G. D., "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[63] H. Stark and J. W. Woods, *Probability and Random Processes with Applications to Signal Processing*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2006.

[64] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. New York, NY: Princeton University Press, 2005.

[65] Power system test case archive. University of Washington. [Online]. Available: http://www.ee.washington.edu/research/pstca/

[66] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 12–19, 2011.

[67] A. Björck, *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[68] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. New York, NY: Chapman and Hall, 1986.

[69] D. W. Scott, *Multivariate Density Estimation: Theory, practice, and Visualization*. New York, NY: Wiley, 1992.

[70] P. Domingos and G. Hulten, "A general framework for mining massive data streams," *Journal of Computational and Graphical Statistics*, vol. 12, no. 4, pp. 945–949, 2003.

[71] T. Duong and M. L. Hazelton, "Cross-validation bandwidth matrices for multivariate kernel density estimation," *Scandinavian Journal of Statistics*, vol. 32, no. 3, pp. 485–506, 2005.

[72] B. A. Turlach, "Bandwidth selection in kernel density estimation: A review," in *CORE and Institut de Statistique*, 1993.

[73] N.-B. Heidenreich, A. Schindler, and S. Sperlich, "Bandwidth selection for kernel density estimation: a review of fully automatic selectors," *AStA Advances in Statistical Analysis*, vol. 97, no. 4, pp. 403–433, 2013.

[74] L. Devroye and G. Lugosi, *Combinatorial Methods in Density Estimation*. New York, NY: Springer, 2001.

[75] C. Heinz and B. Seeger, "Cluster kernels: Resource-aware kernel density estimators over streaming data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 7, pp. 880–893, July 2008.

[76] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[77] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *Computers, IEEE Transactions on*, vol. C-22, no. 11, pp. 1025–1034, Nov 1973.

[78] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[79] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '98, 1998, pp. 73–84.

[80] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug 1999.

[81] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and

densities in noisy, high dimensional data," in *Proceedings of the 2003 SIAM International Conference on Data Mining*, 2003, pp. 47–58.

[82] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[83] A. Mousavi, M. Duckham, R. Kotagiri, and A. Rajabifard, "Spatio-temporal event detection using probabilistic graphical models (PGMs)," in *IEEE Symp. Comput. Intell. and Data Mining*, April 2013, pp. 81–88.

[84] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, Massachusetts: MIT Press, 2001.

[85] F. Zhang, Ed., *The Schur Complement and Its Application*, ser. Numerical Methods and Algorithms. Springer US, 2005, vol. 4.

[86] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863 – 874, 2007.

[87] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[88] K. M. Abadir and J. R. Magnus, *Matrix Algebra*. Princeton, NJ: Princeton University Press, 2008.